Untere Schranken für Steinerbaumalgorithmen und die Konstruktion von Bicliquen in dichten Graphen

DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Herrn Dipl.-Inf. Stefan Kirchner
geboren am 06.04.1978 in Hamburg

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Dr. h.c. Christoph Markschies

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:
Prof. Dr. Wolfgang Coy

Gutachter:

1. PD Dr. Mihyun Kang
2. Prof. Dr. Stefan Hougardy
3. Prof. Dr. Anand Srivastav

eingereicht am:	25. März 2008
Tag der mündlichen Prüfung:	20. Juni 2008

Abstract

This thesis consists of two parts. The first part is concerned with lower bounds for approximating Steiner trees. The Steiner tree problem is to find a shortest subgraph that spans a given set of vertices in a graph and is a classical $\mathcal{NP}$-hard problem. Several approximation algorithms exist, but for most algorithms only lower and upper bounds for the approximation ratio are known. For some of these algorithms we present instances which improve the lower bounds. When the problem is restricted to the class of graphs with k terminals, we also present a method which can be used to determine the approximation ratio of the Relative Greedy Algorithm with arbitrary precision.

The second part is about balanced bicliques, i.e. complete bipartite subgraphs with equal partition sizes. It has been known for a long time that every dense bipartite graph contains a balanced biclique of size $\Omega(\log(n))$, but whether and how such a biclique can be constructed in polynomial time is still unknown. Our contribution to this problem is a polynomial time algorithm which constructs a balanced biclique of size $\Omega(\sqrt{\log(n)})$ in sufficiently large and dense bipartite graphs.

Keywords:
Steiner tree, approximation algorithms, lower bounds, bicliques

Zusammenfassung

Die vorliegende Arbeit besteht aus zwei Teilen. Der erste Teil der Arbeit befasst sich mit unteren Schranken für approximative Steinerbaumalgorithmen. Ein Steinerbaum ist ein kürzester Teilgraph, der eine gegebene Teilmenge der Knoten eines Graphen spannt. Das Berechnen eines Steinerbaumes ist ein klassisches $\mathcal{NP}$-schweres Problem, und es existieren mehrere Approximationsalgorithmen, wobei bei den meisten Algorithmen die Approximationsgüte nur durch untere und obere Schranken eingegrenzt werden kann. Für einige dieser Algorithmen werden in dieser Arbeit Instanzen vorgestellt, welche die unteren Schranken verbessern. Für den Relativen Greedy Algorithmus wird außerdem ein Verfahren vorgestellt, mit dem die Güte des Algorithmus eingeschränkt auf die Graphenklasse mit k Terminalen auf einen beliebigen Faktor genau bestimmt werden kann.

Der zweite Teil der Arbeit widmet sich vollständig bipartiten Subgraphen mit gleicher Partitionsgröße, sogenannten balancierten Bicliquen. Seit langem ist bekannt, dass in dichten bipartiten Graphen balancierte Bicliquen mit $\Omega(\log(n))$ Knoten existieren, aber es ist unbekannt, ob und wie diese in polynomieller Zeit konstruiert werden können. Der zweite Teil liefert dazu einen Beitrag, indem ein polynomieller Algorithmus vorgestellt wird, der in hinreichend großen dichten bipartiten Graphen eine balancierte Biclique mit $\Omega(\sqrt{\log(n)})$ Knoten konstruiert.

Schlagwörter:
Steinerbaum, Approximationsalgorithmen, untere Schranken, Bicliquen

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Teil I

Untere Schranken für Steinerbaumalgorithmen

Kapitel 1

Einleitung

Die Anfänge der Steinerbäume gehen auf Pierre de Fermat (1607–1665) zurück, der sich (unter anderem) mit der Frage beschäftigte, wie die kürzeste Verbindung zwischen drei Punkten in der Ebene aussieht. Im Allgemeinen enthält diese einen Verzweigungspunkt, der heute auch als Fermat- oder Toricellipunkt bekannt ist. Die naheliegende Verallgemeinerung dieser Fragestellung für mehr als drei gegebene Punkte wurde unter anderem von Jakob Steiner (1796–1863) untersucht, der als Mathematikprofessor in Berlin gearbeitet hat und auf dessen Namen die Bezeichnung Steinerbaum zurückgeht.

Das Steinerbaumproblem in der Ebene wurde später auf Graphen verallgemeinert, und allen in dieser Arbeit vorkommenden Steinerbäumen liegt ein Graph zugrunde. Das Steinerbaumproblem in Graphen lässt sich wie folgt formulieren.

> Sei ein Graph $G = (V, E)$ mit einer Kantengewichtsfunktion $w : E \to \mathbb{R}_+$ und einer ausgezeichneten Menge $R \subseteq V$, genannt *Terminale*, gegeben. Bestimme einen kürzesten Teilbaum T von G, der R spannt, d. h. je zwei Knoten aus R sollen durch einen Pfad in T verbunden sein.

Neben der rein mathematischen Fragestellung finden Steinerbäume Anwendungen im VLSI-Design, bei Multicasting-Protokollen und bei phylogenetischen Bäumen aus der Biologie, siehe z. B. [43; 30; 11].

Auf (endlichen) Graphen kann ein minimaler Steinerbaum durch vollständiges Enumerieren gefunden werden. Wie meistens bei $\mathcal{NP}$-schweren kombinatorischen Optimierungsproblemen, zu denen auch das Steinerbaumproblem gehört [41], ist es bei diesem Brute-Force-Ansatz bereits bei kleinen Instanzen aussichtslos, in akzeptabler Zeit eine optimale Lösung zu finden. Es bieten sich dann mindestens zwei Möglichkeiten an. Eine optimale Lösung soll gefunden werden und dafür wird auch exponentielle Laufzeit in

Kauf genommen, jedoch mit einer deutlich besseren Laufzeit als beim Brute-Force-Ansatz. Die einfachsten exakten Algorithmen werden in Abschnitt 1.2 kurz vorgestellt.

Sofern auf eine optimale Lösung verzichtet werden kann und auch eine Näherungslösung ausreicht, sind *Approximationsalgorithmen* eine Alternative. Diese finden in *polynomieller* Zeit eine Lösung, die maximal einen *konstanten* Faktor c von der optimalen Lösung entfernt ist. Die Leitidee hinter Approximationsalgorithmen ist die Folgende: Wenn schon im schlechtesten Fall die Lösung nicht allzu weit vom Optimum entfernt ist, dann sollte sie auf einer „durchschnittlichen Instanz“ deutlich näher an der optimalen Lösung liegen.

Oft fallen jedoch untere und obere Schranken für den Approximationsfaktor c auseinander. Dies trifft auch auf die meisten Approximationsalgorithmen für Steinerbäume zu, die seit Anfang der 90er Jahre entwickelt wurden und für die alle $c < 2$ gilt. Ein Ziel ist dabei, diese zu verbessern, und der Schwerpunkt der vorliegenden Arbeit sind die unteren Schranken. Die Konstruktion einer Instanz oder Instanzenfamilie, welche die untere Schranke verbessert, basiert in der Regel auf der Analyse der oberen Schranke. Dabei wird versucht, eine Instanz zu finden, bei der möglichst viele in der Analyse auftretenden Ungleichungen scharf sind. Gute untere Schranken können dann wiederum Hinweise geben, an welcher Stelle die Analyse – sofern möglich – für die obere Schranke verbessert oder wie der Algorithmus geeignet modifiziert werden kann.

Teil I ist wie folgt aufgebaut. Im weiteren Verlauf des ersten Kapitels folgen die in der Arbeit verwendeten Notationen und die schon erwähnten exakten Algorithmen. Im zweiten Kapitel werden der Relative Greedy Algorithmus, der Loss-Algorithmus und der MSS-Algorithmus vorgestellt, sowie jeweils untere Schranken vorgestellt. Für den Relativen Greedy Algorithmus wird zudem eine Enumerationsmethode vorgestellt, die es gestattet, Worstcase-Instanzen auf einer festen Anzahl an Terminalen beliebig genau zu approximieren.

1.1 Definitionen und Notationen

Im Folgenden werden die in dieser Arbeit verwendeten Notationen und grundlegende Definitionen eingeführt.

Sofern nicht anders beschrieben, bezeichnet $G = (V, E, w)$ stets einen *zusammenhängenden* Graphen mit Knotenmenge V, Kantenmenge E und einer Kantengewichtsfunktion $w : E \to \mathbb{R}_+$. $R \subseteq V$ sei die Terminalmenge von G. Ein Steinerbaum $T = (V', E')$ ist ein zusammenhängender, kreisfreier

Teilgraph von G, der R spannt, d. h. $R \subseteq V'$. Das Gewicht von T sei durch $w(T) := \sum_{e \in E'(T)} w(e)$ definiert, also durch die Summe der beteiligten Kantengewichte. Ein *minimaler* Steinerbaum T^* ist ein Steinerbaum mit minimalem Gewicht, d. h. für alle Steinerbäume T gilt $w(T^*) \leq w(T)$. Ein beliebiger minimaler Steinerbaum aus der Menge aller minimalen Steinerbäume wird mit $SMT(G_R)$[1] notiert und dessen Länge mit $smt(G_R)$. Sofern der zugrunde liegende Graph G bekannt ist, wird stattdessen auch $SMT(R)$ bzw. $smt(R)$ verwendet. Außerdem sei $N(x) := \{y \in V \mid \{x, y\} \in E\}$ die Nachbarschaft von $x \in V$ und die Menge $\{1, \ldots, i\}$ durch $[i]$ abgekürzt. Weitere Notationen werden an geeigneter Stelle eingeführt. In Abbildungen werden Terminale stets durch Quadrate und Nichtterminale durch Kreise dargestellt.

Die beiden folgenden Definitionen beziehen sich auf Approximationsalgorithmen für *Minimierungsprobleme* mit konstanter Güte, da nur diese in Teil I betrachtet werden.[2]

Definition 1 (Approximationsalgorithmus mit konstanter Güte)**.** *Ein Algorithmus A ist ein* Approximationsalgorithmus *mit konstanter Güte, falls A* polynomielle *Laufzeit hat und eine Konstante c existiert mit*

$$\forall I \in \mathcal{I}: \qquad A(I) \leq c \cdot Opt(I). \tag{1.1}$$

Dabei ist $\mathcal{I}$ die Menge aller Instanzen und $Opt(I)$ die Länge einer optimalen Lösung, in diesem Fall also $smt(I)$.

Ein Ziel bei Approximationsalgorithmen besteht darin, Algorithmen mit möglichst kleinem c zu entwickeln. Dies führt auf die *Approximationsgüte* eines Algorithmus.

Definition 2 (Güte eines Approximationsalgorithmus)**.** *Die Güte ρ_A eines Algorithmus A ist definiert durch*

$$\forall I: \qquad \rho_A = \sup_I \frac{A(I)}{Opt(I)}. \tag{1.2}$$

1.2 Exakte Algorithmen für Steinerbäume

Enthält ein Graph G nur wenige Terminale, so kann mittels dynamischer Programmierung in $\mathcal{O}(3^{|R|}|V|^3)$ ein minimaler Steinerbaum berechnet werden [18]. Dieser Algorithmus geht auf Dreyfus und Wagner zurück und wird

[1] Steiner minimum tree

[2] Bei einem Maximierungsproblem muss das Supremum in Definition 2 durch das Infimum ersetzt werden.

als Grundlage für einen effizienteren Algorithmus [25] verwendet, dessen Kerngedanke darin besteht, eine Rekursionsgleichung nicht nur über ein Terminal wie beim Dreyfus-Wagner-Algorithmus zu führen, sondern über mehrere. Dadurch wird eine Verbesserung der Laufzeit auf $\mathcal{O}^*((2+\epsilon)^{|R|})$ für beliebiges $\epsilon > 0$ erreicht.

Ein durch die dynamische Programmierung bedingter angenehmer Nebeneffekt ist, dass zusätzlich alle minimalen Steinerbäume in G mit Terminalmenge $R' \subseteq R$ berechnet werden. Insbesondere sei darauf hingewiesen, dass die beiden Algorithmen polynomielle Laufzeit haben, falls bezüglich $|V|$ nur logarithmisch viele Terminale vorkommen.

Auf der anderen Seite kann ein minimaler Steinerbaum ebenfalls effizient berechnet werden, wenn G nur wenige Nichtterminale enthält. Der Grund liegt in der Beobachtung, dass $\mathrm{SMT}(R)$ gerade ein minimal spannender Baum auf dem induzierten Graphen $[V \setminus A]$ ist, wobei A die Menge der von $\mathrm{SMT}(R)$ nicht verwendeten Nichtterminale ist. Nun kann durch Enumeration der Menge $A \subseteq V \setminus R$ und dem Berechnen eines minimal spannenden Baums auf dem induzierten Graphen $[V \setminus A]$ ein minimaler Steinerbaum für G in $\mathcal{O}(2^{|V|-|R|}|V|^2)$ gefunden werden. Für eine logarithmische Anzahl Nichtterminale hat dieser Algorithmus daher polynomielle Laufzeit.

Ein übliches Vorgehen ist es nun, beide Algorithmen zu kombinieren. Mit dem Ansatz $2^{|R|} = 2^{|V|-|R|}$ ergibt sich für den Fall $|R| = |V|/2$, dass beide Algorithmen nahezu gleiche Laufzeit haben.[3] Indem für $|R| < |V|/2$ der erweiterte Dreyfus-Wagner-Algorithmus und für $|R| \geq |V|/2$ der Enumerationsalgorithmus verwendet wird, kann das Steinerbaumproblem exakt in $2.001^{|V|/2} \ll 1.42^{|V|}$ berechnet werden.

Ein anderer Ansatz besteht darin, das Steinerbaumproblem als ganzzahliges lineares Programm zu modellieren, dieses dann zu relaxieren und mit Schnittebenenverfahren exakt zu lösen [63; 16].

[3] Polynomielle Faktoren in $|V|$ und ϵ seien hier vernachlässigt.

Kapitel 2

Approximationsalgorithmen für Steinerbäume

Das Berechnen eines minimalen Steinerbaumes ist $\mathcal{NP}$-schwer und gehört zu den 21 klassischen Problemen, für die Karp 1972 die $\mathcal{NP}$-Vollständigkeit bewiesen hat [41]. Unter den üblichen komplexitätstheoretischen Annahmen existiert daher kein polynomieller Algorithmus für das Steinerbaumproblem, unabhängig davon gibt es jedoch einige in Polynomialzeit lösbare Spezialfälle. Für $R = V$ ist ein minimaler Steinerbaum identisch zu einem minimal spannenden Baum, für dessen Berechnung es verschiedene Polynomialzeitalgorithmen gibt [10; 48; 54]. Der Fall $|R| = 2$ entspricht gerade einem kürzesten Weg zwischen den zwei Knoten aus R und kann z. B. mit dem Algorithmus von Dijkstra [17] gelöst werden. Desweiteren kann für logarithmisches $|R|$ ebenfalls ein minimaler Steinerbaum in polynomieller Zeit berechnet werden (s. Abschnitt 1.2). Diese drei Steinerbaumklassen dienen als Grundlage für einige Approximationsalgorithmen.

2.1 Resultate zur Nichtapproximierbarkeit

Eine Konsequenz des PCP-Theorem $\mathcal{NP} = \mathcal{PCP}(\mathcal{O}(\ln(n), \mathcal{O}(1))$ [3] ist, dass das Steinerbaumproblem in Graphen nicht beliebig genau approximiert werden kann, sofern $\mathcal{P} \neq \mathcal{NP}$ gilt. Ausgehend von dem PCP-Theorem konnte Thimm [60] unter der Voraussetzung $\mathcal{RP} \neq \mathcal{NP}$ eine Nichtapproximierbarkeitsschranke von $163/162 \approx 1.0062$ angeben, die dann später durch Chlebík und Chlebíková [13] auf $96/95 \approx 1.0105$ verbessert wurde. Bei beiden Arbeiten erfolgt die Reduktion von MAX-E3-LIN2[1].

[1] lineares Gleichungssystem über dem Körper $\mathbb{F}_2$, wobei jede Gleichung genau 3 Variablen hat. Gesucht ist die maximale Anzahl zu erfüllender Gleichungen.

Jahr	**Algorithmus**	**Autoren**	**untere** und **obere Schranken**
1968	MST-Heuristik [27]	Moore	$2 \leq \rho \leq 2$
1993	11/6-Zelikovsky [64]	Zelikovsky	$1.666 \leq \rho \leq 1.834$
1994	Berman-Ramaiyer-Alg. [6]	Berman, Ramaiyer	$1.166 \leq \rho \leq 1.734$
1995	Relativer Greedy Alg. [65]	Zelikovsky	$\mathbf{1.385} \leq \rho \leq 1.694$
1997	RGH-Alg. [42]	Karpinski,Zelikovsky	$1.166 \leq \rho \leq 1.644$
1999	Iterated RGH-Alg. [37]	Hougardy, Prömel	$1.166 \leq \rho \leq 1.598$
2000	MST in Hypergraphs [55]	Prömel, Steger	$1.666 \leq \rho \leq 1.667$
2005	Loss-Algorithmus [57]	Robins, Zelikovsky	$\mathbf{1.223} \leq \rho \leq 1.550$

Tabelle 2.1: *Übersicht über einige Approximationsalgorithmen für das Steinerbaumproblem*

Im Gegensatz dazu sind für einige geometrische Steinerbaumprobleme polynomielle Approximationsschemata bekannt. Dazu gehören das euklidische und rektilineare Steinerbaumproblem in der Ebene [2].

2.2 Übersicht über Approximationsalgorithmen für Steinerbäume

In Tabelle 2.1 sind einige Approximationsalgorithmen für das Steinerbaumproblem abgebildet. Ein von verschiedenen Autoren gewählter Ansatz besteht darin, einen Steinerbaum über einen minimal spannenden Baum in einem geeigneten Graphen zu konstruieren. Dies war für einige Jahre der beste bekannte Approximationsalgorithmus. Anfang der 90er Jahre führte Zelikovsky sogenannte k-Steinerbäume ein (s. Kapitel 2.4), auf deren Grundlage die meisten neueren Algorithmen basieren. Wie aus Tabelle 2.1 ersichtlich weichen die oberen und unteren Schranken der meisten Algorithmen zum Teil erheblich voneinander ab. Für die in der Tabelle fettgedruckten unteren Schranken werden in dieser Arbeit entsprechende Instanzen vorgestellt.

2.3 MST-Heuristik

Im Folgenden wird die MST-Heuristik vorgestellt, welche unabhängig von verschiedenen Autoren gefunden wurde und für die gezeigt werden kann, dass ihre Güte 2 beträgt. Als Hilfskonstruktion dient der sogenannte Terminaldistanzgraph.

Definition 3 (Terminaldistanzgraph)**.** *Seien $G = (V, E)$ mit $w : E \to \mathbb{R}_+$ und $R \subseteq V$ gegeben. Der vollständige Graph (R, E', w') heißt* Terminaldistanzgraph*, falls für die Länge ℓ eines kürzesten Pfades von t_i nach t_j in G die Gleichung $w'(\{t_i, t_j\}) = \ell$ gilt.*

Die MST-Heuristik berechnet einen minimal spannenden Baum im Terminaldistanzgraphen und ersetzt dann die Kanten im Baum durch die korrespondierenden Pfade in G. Sofern dieser Graph nicht kreisfrei ist, werden Kanten aus dem Kreis und Nichtterminale von Grad eins rekursiv entfernt. Ein minimal spannender Baum im Terminaldistanzgraphen wird mit $MST(R_G)$ und seine Länge mit $mst(R_G)$ bezeichnet, wobei jeweils der Index G weggelassen wird, sofern keine Verwechselungsgefahr besteht. Weiterhin gilt $mst(R) \leq 2 \cdot smt(R)$ [27], wobei der Faktor 2 bestmöglich ist. Bei Verwendung geeigneter Datenstrukturen hat die MST-Heuristik eine Laufzeit von $\mathcal{O}(|E| + |V| \ln(|V|))$ [50].

Ein anderer Algorithmus mit der gleichen Güte, der in seinem Ablauf dem Algorithmus von Prim [54] ähnelt, ist in [59] beschrieben.

2.4 k-Steinerbäume

Alle bekannten Steinerbaumalgorithmen mit einer Approximationsgüte echt kleiner zwei basieren auf dem Konzept von sogenannten k-Steinerbäumen, die erstmals von Zelikovsky vorgestellt wurden [64]. Ein Baum mit Blättern B und inneren Knoten K heißt *voll*, wenn $B \subseteq R$ und $K \cap R = \emptyset$. Steinerbäume, die nicht voll sind, können in – nicht notwendig kantendisjunkte – volle Teilbäume zerlegt werden. Diese werden *volle Komponenten* genannt. Die *Größe* einer vollen Komponente ist die Anzahl ihrer Blätter. Falls jede volle Komponente maximal k Terminale enthält, bildet deren Vereinigung einen sogenannten k-Steinerbaum. Ein kürzest möglicher k-Steinerbaum wird als *minimaler k-Steinerbaum* bezeichnet. Als Notation wird dafür $\mathrm{SMT}_k(R)$ verwendet, sowie $\mathrm{smt}_k(R)$ für seine Länge.

Borchers und Du [9] haben für das maximale Verhältnis zwischen der Länge eines minimalen k-Steinerbaums und der Länge eines minimalen Steinerbaums die folgende explizite Formel bewiesen.

Satz 1 (Borchers, Du [9])**.** *Sei $k = 2^r + s$ mit $r \in \mathbb{N}$ und $0 \leq s < 2^r$. Dann gilt*

$$\nu(k) := \sup_I \frac{smt_k(I)}{smt(I)} = \frac{(r+1) \cdot 2^r + s}{r \cdot 2^r + s} \tag{2.1}$$

Eine etwas schwächere Formulierung lautet $\text{smt}_k(G_R) \leq (1 + \frac{1}{\log k}) \cdot \text{smt}(G_R)$. Das Verhältnis zwischen $\text{smt}_k(G_R)$ und $\text{smt}(G_R)$ strebt daher für $k \to \infty$ gegen eins. Einige Algorithmen A aus Tabelle 2.1 approximieren für ein beliebiges konstantes k einen k-Steinerbaum mit Güte ρ_A. Für ein beliebiges $\epsilon > 0$ kann dann ein hinreichend großes k gewählt werden mit $A(G_R) \leq (1 + \epsilon)\rho_A \cdot smt(G_R)$.

Es lässt sich zeigen, dass die MST-Heuristik einen optimalen 2-Steinerbaum berechnet. Nach Satz 1 beträgt die Güte der MST-Heuristik daher zwei. Der Algorithmus von Prömel und Steger ist ein randomisiertes volles Approximationsschema für einen optimalen 3-Steinerbaum. Wegen $\nu(3) = 5/3$ ist dessen Güte $5/3 + \epsilon$.[2] Für $k \geq 4$ ist hingegen das Berechnen eines optimalen k-Steinerbaums $\mathcal{APX}$-hart [7], was durch eine Reduktion von Node-Cover mit Maximalgrad drei gezeigt werden kann.

2.5 Relativer Greedy Algorithmus

Ein grundlegendes Konzept einiger Approximationsalgorithmen besteht darin, einen k-Steinerbaum zu berechnen, indem iterativ solange auf maximal k Terminalen ein Steinerbaum berechnet wird, bis deren Vereinigung einen k-Steinerbaum für den gesamten Graphen bildet.

Die Algorithmen unterscheiden sich dabei hinsichtlich der Wahl der Komponenten. Dazu dient in der Regel eine Bewertungsfunktion, die im folgenden Auswahlfunktion genannt wird und jeder maximal k-elementigen Menge einen Wert zuweist, wie „gut“ diese ist.

Im Folgenden wird der Relative Greedy Algorithmus beschrieben, danach werden untere Schranken für diesen Algorithmus gezeigt und anschließend ein Enumerationsverfahren zum Konstruieren von unteren Schranken vorgestellt.

2.5.1 Beschreibung des Algorithmus

Der Relative Greedy Algorithmus berechnet zu einem gegebenen Graphen $G = (V, E, w)$, $R \subseteq V$ und von G unabhängigem konstanten $k \in \mathbb{N}$ einen k-Steinerbaum. Dabei wählt er sich in jedem Schritt eine Menge mit maximal k Terminalen, berechnet darauf einen minimalen Steinerbaum und iteriert das solange, bis diese die Terminalmenge spannen.

Im Abschnitt 2.3 wurde bereits der Terminaldistanzgraph definiert. Für den Relativen Greedy Algorithmus wird eine Modifikation benötigt.

[2] Exakt 5/3, falls die Kodierungslänge für jedes Kantengewicht konstant ist.

Definition 4. *Seien $t_1, \dots t_i$ Teilmengen von Terminalen. Die Notation $mst(R/t_1, \dots t_i)$ bezeichne die Länge eines minimal spannenden Baumes im Terminaldistanzgraph, bei dem für jedes t_j mit $j \in [i]$ zwischen je zwei Knoten aus t_j eine Kante mit Gewicht 0 hinzugefügt wird.*

Für eine Menge t von Terminalen ist die Differenz $mst(R/t_1, \dots t_i) - mst(R/t_1, \dots t_i, t)$ gerade die Längenänderung des minimal spannenden Baumes im Terminaldistanzgraphen, wenn Nullkanten zwischen je zwei Knoten aus t hinzugefügt werden.

Der Relative Greedy Algorithmus wählt in jeder Iteration gerade ein Tupel t_{i+1} mit $|t_{i+1}| \leq k$, welches diese Differenz im Verhältnis zum $smt(t_{i+1})$ maximiert, d. h. die Funktion

$$f_i(t) := \frac{smt(t)}{mst(R/t_1, \dots, t_i) - mst(R/t_1, \dots, t_i, t)} \tag{2.2}$$

minimiert. Der Algorithmus gibt die Vereinigung der minimalen Steinerbäume auf den Teilmengen t_i als volle Komponenten eines k-Steinerbaumes zurück. Analog zum SMT_k sei RGA_k der vom Relativen Greedy Algorithmus berechnete k-Steinerbaum und rga_k dessen Länge. Der Pseudocode ist in Abbildung 2.1 dargestellt.

Relativer Greedy Algorithmus

Input: $G = (V, E, w)$, $R \subseteq V$
Output: Ein Steinerbaum mit einer Länge maximal $(1 + \ln(2)) \cdot smt_k(G_R)$
1 $i := 0$
2 **while** $mst(R/t_1, \dots, t_i) \neq 0$
3 wähle $t_{i+1} \subseteq R, |t_{i+1}| \leq k$, das
4 $f_i(t_{i+1}) := \frac{smt(t_{i+1})}{mst(R/t_1,\dots,t_i) - mst(R/t_1,\dots,t_{i+1})}$ minimiert
5 $i := i + 1$
6 $i_{max} := i$
7 **return** $\bigcup_{i=1}^{i_{max}} SMT(t_i)$

Abbildung 2.1: *Relativer Greedy Algorithmus*

Das Einfügen von Nullkanten pro gewählter Komponente t_i im Terminaldistanzgraphen führt dazu, dass für den Algorithmus in den folgenden Schritten stets eine die Auswahlfunktion minimierende Terminalmenge t existiert, bei denen keine zwei Terminale $\{v, w\} \subset t$ im aktuellen Distanzgraphen durch einen Pfad der Länge null verbunden sind. Andernfalls gilt

$f_i(t) \geq f_i(t \setminus \{v\})$, da der Nenner der Auswahlfunktion unverändert bleibt und $smt(t) \geq smt(t \setminus \{v\})$ gilt. Es genügt daher, nur Teilmengen von Terminalen zu betrachten, die keine zwei Terminale enthalten, die im aktuellen Terminaldistanzgraphen durch einen Pfad der Länge Null verbunden sind. Unter dieser Voraussetzung gilt für je zwei Mengen $t_i, t_j \in t_1, \ldots, t_{i_{max}}$ mit $i \neq j$ die Ungleichung $|t_i \cap t_j| \leq 1$, d. h. sie schneiden sich maximal in einem Terminal.

Technische Anmerkung: Die Abbruchbedingung der while-Schleife in Abbildung 2.1 garantiert, dass das Ergebnis, welches der Algorithmus berechnet, alle Terminale spannt. Dieses ist aber – wie bei der MST-Heuristik – nicht notwendig kreisfrei. Die Kreise können z. B. dadurch zerstört werden, dass in einem Postprocessing ein minimal spannender Baum in RGA_k berechnet wird. Im Folgenden sei der vom Relativen Greedy Algorithmus berechnete Graph ohne Beschränkung der Allgemeinheit kreisfrei.

Zelikovsky hat für den Relativen Greedy Algorithmus eine obere Schranke für die Approximationsgüte bewiesen.

Satz 2 (Zelikovsky, [65])**.** *Für jedes* $2 \leq k \in \mathbb{N}$ *und jeden Graphen* G *mit Terminalmenge* R *gilt*

$$rga_k(G_R) \leq (1 + \ln(2)) \cdot smt_k(G_R). \tag{2.3}$$

Aus Satz 1 folgt, dass für hinreichend großes k der Relative Greedy Algorithmus einen Steinerbaum berechnet, dessen Länge maximal einen Faktor von 1.694 von $smt(R)$ entfernt ist.

2.5.2 Untere Schranken für den Relativen Greedy Algorithmus

In diesem Abschnitt werden untere Schranken für den Relativen Greedy Algorithmus vorgestellt. Eine gute Übersicht über untere Schranken für verschiedene Steinerbaumalgorithmen findet sich in [29]. Speziell für den Relativen Greedy Algorithmus ist dort eine untere Schranke von 4/3 bewiesen worden. Die verwendete Instanz hat eine Gitterstruktur und ist relativ kompliziert. Eine deutlich vereinfachte Instanz, ebenfalls mit einer Güte von 4/3, befindet sich in [45] und wurde in der selben Arbeit auf 1.37 erweitert. Im Folgenden wird ein ähnlicher Ansatz beschrieben, der in [36] verwendet wird.

Die prinzipielle Idee, um gute untere Schranken für den Relativen Greedy Algorithmus zu finden, ist die folgende. In jedem Schritt sollte die Komponente, die der Algorithmus wählt, jede Komponente des optimalen Steinerbaums schneiden. Dadurch wird die optimale Komponente in jedem Schritt

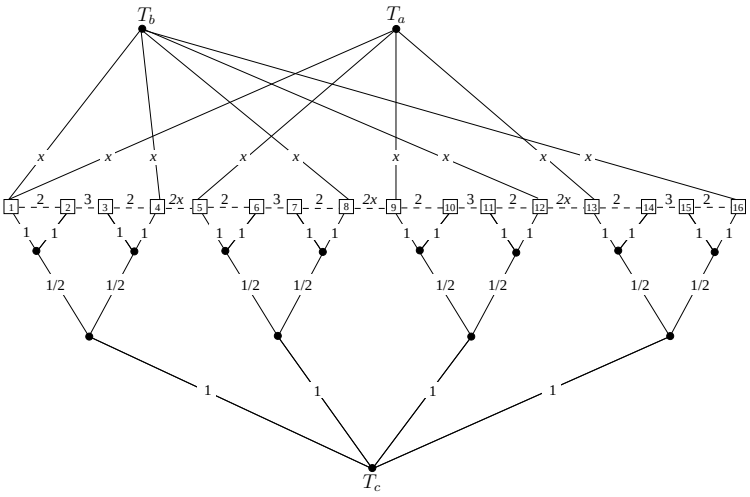


Abbildung 2.2: *Instanz zur unteren Schranke* $\frac{41}{30} - \epsilon$ *(*G_k *mit* $k = 4$*)*

uninteressanter für den Algorithmus. Diese Vorgehensweise führt dann zu Strukturen, bei denen die Terminale in einem Gitter angeordnet werden können, so dass der minimale Steinerbaum aus horizontalen Komponenten und der vom Algorithmus berechnete Steinerbaum aus vertikalen Komponenten besteht.

2.5.2.1 Instanz mit einer Güte von $\frac{41}{30} - \epsilon$

Die in Abbildung 2.2 veranschaulichte Instanz G_k enthält $4k$ Terminale $w_1, \ldots, w_{4k}$, drei spezielle Knoten T_a, T_b, T_c, $2k$ Knoten $y_1, \ldots, y_{2k}$ und k Knoten $z_1, \ldots, z_k$. Jedes Terminal w_{4i-3} ist mit T_a verbunden, jedes Terminal w_{4i} und das Terminal w_1 ist mit T_b mit einer Kante mit Gewicht x verbunden. Außerdem ist jeder Knoten y_i mit den Terminalen w_{2i-1} und w_{2i} durch Kanten mit Gewicht 1 verbunden, die Knoten z_i sind mit y_{2i-1} und y_{2i} durch Kanten mit Gewicht $1/2$ und die Knoten z_i sind mit T_c durch Kanten der Länge 1 verbunden für $1 \leq i \leq k$. Jedes Terminal w_{2i-1} ist mit w_{2i} durch eine Kante der Länge 2 verbunden mit $1 \leq i \leq 2k$, w_{4i+2} ist mit w_{4i+3} durch eine Kante der Länge 3 verbunden mit $0 \leq i \leq k-1$ und w_{4i} ist mit w_{4i+1} durch eine Kante der Länge $2x$ verbunden für $1 \leq i \leq k-1$.

Satz 3 (Hougardy, K. [36])**.** *Für jedes* $\epsilon > 0$ *und für ein geeignetes* $x = x(\epsilon)$ *gilt*

$$\lim_{k \to \infty} \frac{rga_{4k}(G_k)}{smt_{4k}(G_k)} = \frac{41}{30} - \frac{\epsilon}{3}.$$

Beweis. Der Wert von x wird derart bestimmt, dass der Relative Greedy Algorithmus in den ersten beiden Schritten die Terminalmengen $t_a := N(T_a)$ und $t_b := N(T_b)$ wählen kann. $SMT(t_a)$ und $SMT(t_b)$ sind die beiden Sterne mit Mittelpunkt T_a bzw. T_b. In allen weiteren Schritten ist dann keine Einsparung gegenüber $MST(R/t_a, t_b)$ mehr möglich, so dass nur noch volle Komponenten der Größe zwei gewählt werden. Der minimale Steinerbaum besteht aus dem Binärbaum mit der Wurzel T_c und hat die Länge $6k$. Mit t_c wird die Menge aller Terminale in G_k bezeichnet. Im Folgenden sei $2 < x < 2.5$. Dann bilden die Kanten $\{u, v\}$ mit $\{u, v\} \subset R$ einen minimal spannenden Baum im Terminaldistanzgraphen mit der Länge $7k + (k-1) \cdot 2x$. In Abbildung 2.2 sind das gerade die horizontal gezeichneten Kanten. Damit im ersten Schritt t_a die Auswahlfunktion minimiert, muss zum einen die Ungleichung

$$f_0(t_a) = \frac{kx}{(k-1) \cdot 2x} \leq \frac{(k+1)x}{(k-1) \cdot 2x + 3} = f_0(t_b) \tag{2.4}$$

gelten, was für $k \geq \frac{2x}{2x-3}$ der Fall ist. Zusätzlich muss die Ungleichung

$$f_0(t_a) = \frac{kx}{(k-1) \cdot 2x} \leq \frac{6k}{7k + (k-1) \cdot 2x} = f_0(t_c) \tag{2.5}$$

erfüllt sein, was für $k \geq \frac{12-2x}{5-2x}$ gilt. Weiterhin existiert keine weitere Teilmenge von Terminalen, deren Auswahlfunktion einen kleineren Wert als $f_0(t_a)$ hat. Daher minimiert t_a im ersten Schritt die Auswahlfunktion des Relativen Greedy Algorithmus.

Im zweiten Schritt genügt es aufgrund von Symmetrie und Monotonie der Funktion f_i, die vier Teilmengen von Terminalen $t_1 = \{w_1, w_2, w_3, w_4\}$, $t_2 = t_1 \cup \{w_{4j-1} | 2 \leq j \leq k\} \cup \{w_{4j} | 2 \leq j \leq k\}$, $t_3 = t_2 \cup \{w_{4j-2} | 2 \leq j \leq k\}$ und t_b zu betrachten.

Dabei wird die Auswahlfunktion von t_b minimiert, falls

$$\begin{aligned} f_1(t_b) = \frac{(k+1)x}{3k} &\leq \min\left(f_1(t_1), f_1(t_2), f_1(t_3)\right) \\ &= \min\left(\frac{5}{7}, \frac{\frac{7}{2} \cdot k + \frac{5}{2}}{5 \cdot k + 2}, \frac{5 \cdot k + 1}{7 \cdot k}\right) \\ &= \frac{7k+5}{10k+4} \qquad \text{für } k \geq 15 \end{aligned} \tag{2.6}$$

gilt.

Für $0 < \epsilon < 0.1$, $x := \frac{21}{10} - \epsilon$ und hinreichend großes k sind die Ungleichungen (2.4), (2.5) und (2.6) erfüllt. Außerdem ist, wie gefordert, $2 < x < 2.5$. Daher kann der Relative Greedy Algorithmus in den ersten beiden Schritten

die Terminalmengen t_a und t_b wählen. In allen weiteren Schritten $i \geq 3$ gilt $f_{i-1}(t) \geq 1$ für alle Terminalmengen t, wobei Gleichheit für $t = \{w_{2j-1}, w_{2j}\}$ mit $j \in [2k]$, $t = \{w_{4j-4}, w_{4j-3}, w_{4j-2}\}$ und $t = \{w_{4j-3}, w_{4j-2}, w_{4j-1}\}$ jeweils mit $j \in [k]$ zutrifft. Es kann daher davon ausgegangen werden, dass nur noch volle Komponenten der Größe zwei gewählt werden. Daher beträgt die Gesamtlänge des vom Algorithmus berechneten Steinerbaums

$$k \cdot x + (k+1) \cdot x + 2k \cdot 2 = 4k + x + 2kx. \tag{2.7}$$

Für $x = \frac{21}{10} - \epsilon$ ergibt sich damit das Verhältnis

$$\frac{rga_{4k}(G_k)}{smt_{4k}(G_k)} = \frac{4k + \frac{21}{10} - \epsilon + 2k(\frac{21}{10} - \epsilon)}{6k} = \frac{41}{30} - \frac{\epsilon}{3} + \frac{21 - 10\epsilon}{60k}, \tag{2.8}$$

welches für $k \to \infty$ gegen $\frac{41}{30} - \frac{\epsilon}{3}$ strebt. □

2.5.2.2 Instanz mit einer Güte von 1.38538

In diesem Abschnitt wird eine Instanz $G_{\ell,k}$ für eine verbesserte untere Schranke vorgestellt. Die Grundidee basiert darauf, die optimale Lösung aus der ersten Instanz in eine Gitterstruktur einzubetten.

Die Instanz $G_{\ell,k}$ besteht aus einem $\ell \times 4k$-Terminalgitter, bei der das Terminal w_{4k} von G_k von jeder Reihe zu einem einzigen Terminal verschmolzen werden (das Terminal rechts oben in Abbildung 2.3). Für jede Reihe von Terminalen wird der Graph $G_k - T_a - T_b$ hinzugefügt, während die Terminale von jeder ungeraden Spalte (mit Ausnahme der Spalte $4k-1$) und dem identifizierten Terminal alternierend durch Sterne mit Mittelpunkt B_i $(0 \leq i \leq k-1)$ und A_i $(0 \leq i \leq k-2)$ verbunden sind.

Ähnlich wie bei der ersten Instanz werden einige Kantengewichte fest gesetzt und andere zunächst variabel gehalten. Diese werden dann später unter bestimmten Nebenbedingungen geeignet gewählt, um das Verhältnis der Algorithmuslösung zur Länge des minimalen Steinerbaums zu maximieren.

Die Kanten, welche denen in G_k zu T_c inzidenten Kanten entsprechen, haben in $G_{k,\ell}$ das Gewicht y und die entsprechenden Kanten mit Gewicht $1/2$ in G_k haben das Gewicht x. Die Gewichte $a_i = a_i(k, \ell, x, y)$ und $b_i = b_i(k, \ell, x, y)$ der Kanten, die zu den Knoten A_i bzw. B_i inzident sind, werden später genauer bestimmt. Alle übrigen Kanten haben das Gewicht eins. Aus technischen Gründen seien die Folgen (a_i) und (b_i) monoton wachsend. Außerdem gelte $a_{k-2} < b_0$ und $a_0 > 1 + x + y$.

Ziel dieser Konstruktion ist es, dass der Relative Greedy Algorithmus nacheinander alle $N(A_i)$ und $N(B_i)$ wählt. $SMT(N(A_i))$ und $SMT(N(B_i))$ sind dabei die Sterne mit Mittelpunkten A_i bzw. B_i Anschließend werden nur

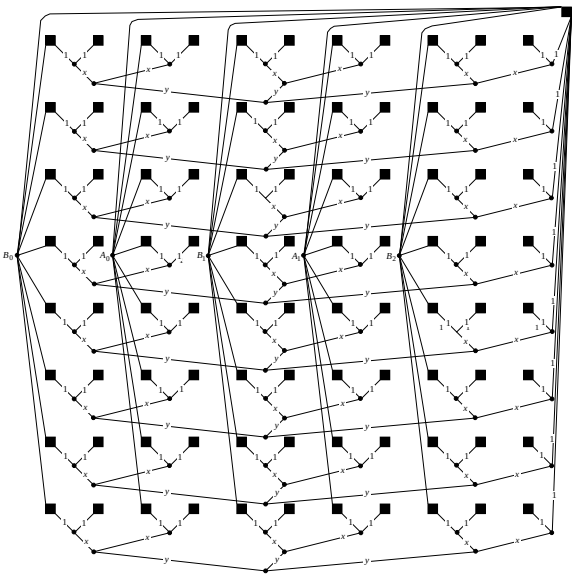


Abbildung 2.3: *Verbesserte untere Schranke* ($G_{k,\ell}$ *mit* $k = 3, \ell = 8$)

noch volle Komponenten der Größe zwei gewählt. Der minimale Steinerbaum besteht hingegen aus den horizontalen Bäumen, d. h. aus dem Teilgraphen $G_{\ell,k} \setminus (B_{k-1} \cup \bigcup_{i=0}^{k-2}(A_i \cup B_i))$.

Satz 4 (Hougardy, K. [36])**.** *Die Approximationsgüte für den Relativen Greedy Algorithmus ist maximal* 1.38538*.*

Beweis. Wegen der Voraussetzung $b_j > a_i > 1 + x + y$ bestehen die korrespondierenden kürzesten Pfade in $G_{k,\ell}$ des minimal spannenden Baumes im Terminaldistanzgraphen nur aus den Kanten der horizontalen Komponenten. Daher ergibt sich

$$mst(R) = ((6 + 2x)k + 2(1 + x + y)(k - 1))\,\ell. \tag{2.9}$$

Nachdem $N(A_i)$ gewählt wurde, beträgt die Änderung der Länge im minimal spannenden Baum $2(1+x+y)\ell$. Nun muss eine Teilmenge von Terminalen innerhalb einer Reihe gefunden werden, die die Funktion f_i minimiert, nachdem die Terminalmengen $N(A_0), \ldots, N(A_{i-1})$ ausgewählt wurden. Für ein fest gewähltes h mit $1 \le h \le \ell$ sei $S_h = \{w_{h,i} | 1 \le i \le 4k\}$. Unter Berücksichtigung von Symmetrie und Monotonie der Funktion f_i im $(i+1)$-ten Schritt genügt es, die Teilmengen $S_h^{i,1} = S_h \setminus (\{w_{h,4j-1} | 1 \le j \le i\} \cup \{w_{h,4j} | 1 \le j \le i\})$ und $S_h^{i,2} = \{w_{h,j} | 4i + 1 \le j \le 4k\}$ zu betrachten.

Die Terminalmenge $N(A_i)$ minimiert die Auswahlfunktion im Schritt $i+1$,

falls

$$\begin{aligned} f_i(N(A_i)) &= \frac{\ell+1}{\ell} \cdot \frac{a_i(k,\ell,x,y)}{2(1+x+y)} \\ &\leq \min(f_i(S_h^{i,1}), f_i(S_h^{i,2})) \\ &= \min\left(\frac{(4+2x+y)k-(2+x)i}{\frac{mst(R)}{\ell}-(4+2x)i}, \right. \\ &\qquad \left. \frac{(k-i)(4+2x+y)}{(k-i)(6+2x)+(k-i-1)(2+2x+2y)} \right) \\ &= f_i(S_h^{i,1}). \end{aligned} \tag{2.10}$$

gilt. Durch den Ansatz $f_i(N(A_i)) = f_i(S_h^{i,1})$ ergibt sich für a_i der Wert

$$a_i(k,\ell,x,y) = \frac{\ell}{\ell+1} \cdot \frac{2(1+x+y)\left((4+2x+y)k-(2+x)i\right)}{\frac{mst(R)}{\ell}-(4+2x)i}, \tag{2.11}$$

wobei die Folge a_i monoton wächst.

Nachdem alle Terminalmengen $N(A_i)$ gewählt wurden, verläuft die Auswahl der Terminalmengen $N(B_i)$ ähnlich. Nachdem $N(B_i)$ gewählt wurde, beträgt die Änderung des minimal spannenden Baumes $(2+2x)\ell$.

Es muss wieder die Teilmenge an Terminalen innerhalb einer Reihe gefunden werden, die die Funktion f_{k-1+i} minimiert, nachdem die Terminalmengen $N(A_\nu)$ mit $0 \leq \nu \leq k-2$ und $N(B_j)$ mit $j < i < k-1$ gewählt wurden.

Für den Schritt $k+i$ genügt es unter Berücksichtigung von Symmetrie und Monotonie der Funktion f_{k-1+i} die Teilmenge

$$S_h^{i,3} = \{w_{h,4j+\kappa} | i \leq j < k, 1 \leq \kappa \leq 2\} \cup \{w_{h,4k-1}, w_{h,4k}\}. \tag{2.12}$$

zu betrachten. Im $(k+i)$-ten Schritt minimiert $N(B_i)$ die Auswahlfunktion, falls

$$\begin{aligned} f_{k-1+i}(N(B_i)) &= \frac{\ell+1}{\ell} \cdot \frac{b_i(k,\ell,x,y)}{2+2x} \\ &\leq f_{k-1+i}(S_h^{i,3}) \\ &= \frac{(k-i)(2+x+y)+2+x}{(k-i)(4+2x)+2} \end{aligned} \tag{2.13}$$

gilt. Analog wie bei a_i erhält man durch Gleichsetzen für b_i den Wert

$$b_i(k,\ell,x,y) = \frac{\ell}{\ell+1} \cdot 2(1+x) \cdot \frac{(k-i)\cdot(2+x+y)+2+x}{(k-i)\cdot(4+2x)+2}. \tag{2.14}$$

Eine Ausnahme in der Berechnung bildet die als letzte gewählte vertikale Terminalmenge $N(B_{k-1})$. Die dazu konkurrierende Komponente $S_h^{k-1,3}$ besteht nur aus vier Terminalen, und es gibt keine Kante mit Kosten y in $SMT(S_h^{k-1,3})$. Daher gilt

$$b_{k-1}(\ell, x) = \frac{\ell}{\ell+1} \cdot 2(1+x) \cdot \frac{2+x}{3+x}. \tag{2.15}$$

Analog zu (a_i) ist auch (b_i) monoton wachsend.

Nachdem alle $N(A_i)$ und $N(B_i)$ vom Algorithmus gewählt wurden, soll keine weitere Einsparung zum Distanzgraphen mehr möglich sein, so dass der Algorithmus nur noch Komponenten der Größe zwei wählt. Dies wird durch die Bedingung

$$4k < \min((2+2x+y)k+1, (3+2x)k) \tag{2.16}$$

erreicht. Andernfalls wählt der Algorithmus die Terminalmengen $\{w_{h,2i} | 1 \leq i \leq 2k\} \cup \{w_{h,4k-1}\}$ bzw. $\{w_{h,4i+j} | 0 \leq i < k, 2 \leq j \leq 4\}$.

Die im Beweis benutzten Nebenbedingungen sind $a_0 > 1 + x + y$, $\max\{a_{k-2}\} < \min\{b_0\}$, das monotone Wachstum von a_i und b_i, sowie (2.16). Sofern jede dieser Nebenbedingungen erfüllt ist, ergibt sich damit eine untere Schranke für die Approximationsgüte von

$$\frac{4k\ell + (\ell+1)\sum_{i=0}^{k-2} a_i + (\ell+1)\sum_{i=0}^{k-1} b_i}{(4+2x+y)k\ell}. \tag{2.17}$$

Der letzte Schritt besteht darin, den Variablen derart Werte zuzuweisen, dass unter Berücksichtigung der Nebenbedingungen der Term (2.17) maximiert wird. Dies ist der Fall für $k = 5$ und $\ell \geq 15$ und den numerisch ermittelten Werten $x = 0.8688$ und $y = 0.0630$. Insgesamt wird damit eine untere Schranke von 1.38538 für die Approximationsgüte des Relativen Greedy Algorithmus erreicht. □

Die Tabelle 2.2 zeigt die einzelnen Werte für a_i und b_i.

i	0	1	2	3	4
a_i	1.9403	1.95562	1.97506	2.00054	—
b_i	2.00125	2.04984	2.1273	2.27071	2.59829

Tabelle 2.2: *Gewichte für die vertikalen Sterne mit einer Genauigkeit von fünf Stellen ($k = 5, \ell \geq 15, x = 0.8688$ und $y = 0.0630$)*

Mit einer Implementation des Relativen Greedy Algorithmus von Hanke [31] konnte bei Eingabe der vorliegenden Instanz die ermittelte untere Schranke verifiziert werden.

2.5.3 Worstcase-Instanzen für den Relativen Greedy Algorithmus

Es zeigt sich, dass bereits auf wenigen Terminalen Instanzen existieren, bei welchen der Algorithmus relativ weit vom Optimum entfernt ist. Dies kann ein Hinweis sein, dass die prinzipiellen Schwächen des Algorithmus bereits auf kleinen Instanzen erkennbar werden. Umgekehrt könnten dann gute untere Schranken auf kleinen Graphen eventuell geeignet kombiniert werden, um bessere untere Schranken auf größeren Strukturen zu erhalten. Dieses Vorgehen hat sich jedenfalls bei dem Graphen aus Abbildung 2.3 bewährt, der aus einer geeigneten Kombination des Graphen in Abbildung 2.2 entstanden ist, welche wiederum eine Erweiterung aus [45] ist.

In diesem Abschnitt ist der Fokus nicht auf einzelne Instanzen ausgerichtet, sondern es soll der Frage nachgegangen werden, wie eine Instanz auf k Terminalen aussieht, bei der die Lösung des Relativen Greedy Algorithmus das schlechteste Verhältnis zum minimalen Steinerbaum hat.

Ziel dieses Abschnittes ist es, eine möglichst kleine Klasse $\mathcal{G}_k$ **ungewichteter** Graphen auf k Terminalen mit folgender Eigenschaft zu erzeugen: Es gibt ein $G^* \in \mathcal{G}_k$ und eine geeignete Kantengewichtsfunktion $w^* : E(G^*) \to \mathbb{R}_+$, so dass für jeden Graphen G auf k Terminalen $rga(G_R)/smt(G_R) \leq rga(G^*_R)/smt(G^*_R)$ gilt. (G^*, w^*) wird im folgenden auch **Worstcase-Instanz** (auf k Terminalen) genannt, sowie die Elemente von $\mathcal{G}_k$ als **Topologie** bezeichnet.

Dieses Vorhaben kann in drei Teile zerlegt werden, auf die im Folgenden näher eingegangen wird:

1. Charakterisierung einer endlichen Menge $\mathcal{G}_k$
2. Enumeration aller dieser Topologien
3. Das Bestimmen einer geeigneten Kantengewichtsfunktion zu jedem $G \in \mathcal{G}_k$

2.5.3.1 Charakterisierung einer endlichen Menge $\mathcal{G}_k$

Die folgenden Lemmata gestatten es, einige Strukturaussagen für $G \in \mathcal{G}_k$ zu treffen, wodurch die Enumeration an Topologien eingeschränkt werden kann. Es sei zunächst angenommen, dass $\mathcal{G}_k \neq \emptyset$ gilt. Diese Annahme wird später im Abschnitt 2.5.3.3 gerechtfertigt.

Lemma 1. *Es gibt eine Worstcase-Instanz, bei der jedes Nichtterminal entweder in $SMT(R)$ oder in $RGA(R)$ enthalten ist.*

Beweis. Sei (G^*, w^*) eine Worstcase-Instanz, wobei W die Menge der von $SMT(R)$ und $RGA(R)$ gemeinsam verwendeter Nichtterminale ist und $|W|$ minimal ist. Füge nun Kanten zwischen je zwei Terminalen mit dem Gewicht aus dem Terminaldistanzgraphen ein und lösche alle Nichtterminale, die weder in $SMT(R)$ noch in $RGA(R)$ vorkommen. Durch diese Transformation ändert sich weder $SMT(R)$ noch $RGA(R)$. Für $W = \emptyset$ ist die Behauptung gezeigt, andernfalls sei $v \in W$. Füge in G^* ein Nichtterminal v' mit $N(v') := N(v)$ und $w^*(\{v', x\}) = w^*(\{v, x\})$ für alle $x \in N(v)$ hinzu. Ändere nun $SMT(R)$ entsprechend, so dass statt v das Nichtterminal v' benutzt wird. Die Länge bleibt unverändert, aber es werden nur noch $|W| - 1$ gemeinsame Nichtterminale mit $RGA(R)$ benutzt, im Widerspruch zur Wahl von W. □

Lemma 2. *Es gibt eine Worstcase-Instanz, bei der alle Nichtterminale den Grad drei haben.*

Beweis. Sei G^* eine Instanz, die das vorherige Lemma erfüllt. Sofern ein Steinerbaum T ein Nichtterminal v mit Grad zwei enthält, können in G^* die Kanten $\{v, x\}, \{v, y\} \subseteq E(T)$ entfernt und die Kante $\{x, y\}$ mit Kantengewicht $w(\{v, x\})+w(\{v, y\})$ eingefügt werden. T wird entsprechend modifiziert und v entfernt.

Angenommen es gibt ein Nichtterminal v von Grad mindestens vier. Dann kann seine Nachbarschaft in zwei disjunkte Mengen $N_1(v)$ und $N_2(v)$ mit $|N_i(v)| \geq 2$ für $i \in \{1, 2\}$ zerlegt werden. Füge in G^* nun die Knoten v_1 und v_2 hinzu und verbinde v_i mit $N_i(v)$ für $i \in \{1, 2\}$, wobei das Kantengewicht jeweils übernommen wird, d. h. $w(\{v_i, x\}) := w(\{v, x\})$ für $x \in N(v_i)$. Füge zusätzlich zwischen v_1 und v_2 eine Kante mit Gewicht null ein und lösche v (s. Abbildung 2.4, links).

Die Knoten v_1 und v_2 haben jetzt einen Grad von mindestens drei und höchstens $|N(v)| - 1$. Die Länge von $SMT(R)$ und der Wert der Auswahlfunktion für eine beliebige Terminalmenge ändert sich dadurch nicht. Dieses Verfahren kann so lange iteriert werden, bis alle Nichtterminale den Grad drei haben. □

Lemma 3. *Es gibt eine Worstcase-Instanz, bei welcher der $SMT(R)$ aus genau einer vollen Komponente besteht, d. h. jedes Terminal von $SMT(R)$ ist ein Blatt.*

Beweis. Angenommen ein optimaler Steinerbaum T besteht aus mehreren vollen Komponenten. Sei t ein Terminal, das kein Blatt in T ist (s. Abbildung 2.4, rechts). Dann gibt es zwei zu t inzidente Kanten $\{t, x\}, \{t, y\} \in E(T)$ mit Gewicht a bzw. b. Füge jetzt einen neuen Knoten v und die Kanten $\{x, v\}$, $\{y, v\}$ und $\{t, v\}$ mit Kantengewichten a, b bzw. null ein und lösche

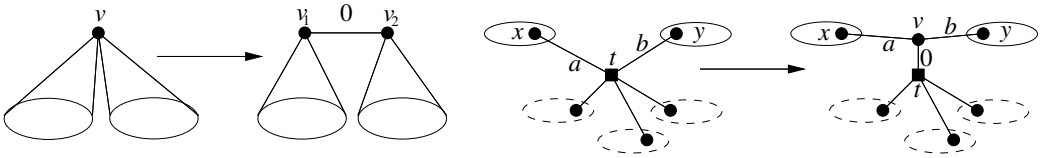


Abbildung 2.4: *(a) Skizze zu Lemma 2; (b) Skizze zu Lemma 3*

$\{t, x\}, \{t, y\}$. Die Länge von T ändert sich dadurch nicht, die Zahl der vollen Komponenten reduziert sich jedoch um eins. Mehrmaliges Iterieren liefert eine Instanz, bei der die optimale Lösung aus genau einer vollen Komponente besteht. □

Korollar 1. *Indem die Transformationen aus den drei Lemmata hintereinander ausgeführt werden, ergibt sich eine Instanz G^*, die alle drei Lemmata erfüllt.*

2.5.3.2 Enumeration aller Topologien mit k Terminalen

Enumeration aller Bäume mit inneren Knoten von Grad drei

Nach Lemma 2 gibt es eine Worstcase-Instanz, bei der alle Nichtterminale aus $SMT(R)$ und $RGA(R)$ den Grad drei haben. Im Folgenden wird daher ein Enumerationsverfahren beschrieben, das die Menge $\mathcal{T}_n$ aller *ungelabelten* Bäume mit n Blättern, bei denen alle inneren Knoten den Grad drei haben, erzeugt. Für $|\mathcal{T}_n|$ ist eine rekursive Formel bekannt [14].[3]

Lemma 4. *Sei $T \in \mathcal{T}_n$. Dann besitzt T einen inneren Knoten, der (mindestens) zu zwei Blättern adjazent ist.*

Beweis. Sei i die Anzahl innerer Knoten von T. Dann gelten die beiden Gleichungen $i + n = |V(T)|$ und $3i + n = 2|E(T)| = 2|V(T)| - 2$. Aufgelöst ergibt sich $i = |V(T)|/2 - 1$ und $n = |V(T)|/2 + 1$, weshalb $|V(T)|$ gerade ist. Da jedes Blatt genau zu einem inneren Knoten adjazent ist und es mehr Blätter als innere Knoten gibt, existieren zwei Blätter, die zu dem gleichen inneren Knoten adjazent sind. □

Dieses Lemma gestattet es nun, solche Bäume rekursiv zu konstruieren. Sei $T \in \mathcal{T}_n$ mit $n \geq 2$ und seien b_1 und b_2 zwei Blätter von T, die zu dem gleichen Knoten adjazent sind. Dann ist $T - b_1 - b_2 \in \mathcal{T}_{n-1}$ und hat ein Blatt weniger als T.

Umgekehrt kann also aus $\mathcal{T}_n$ die Menge $\mathcal{T}_{n+1}$ gebildet werden, indem für jedes $T \in \mathcal{T}_n$ und jedes Blatt b aus T zwei weitere Blätter b_1 und b_2 an b

[3] `http://www.research.att.com/~njas/sequences/A000672`

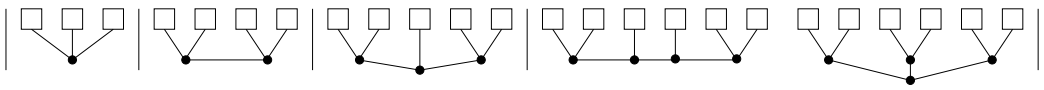


Abbildung 2.5: $\mathcal{T}_n$ *für* $n \in \{3, 4, 5, 6\}$

angehängt werden und in der dadurch entstandenen Menge isomorphe Bäume entfernt werden. Die Mengen $\mathcal{T}_n$ für $3 \leq n \leq 6$ sind in Abbildung 2.5 dargestellt.

Sind bei einem Baum zwei Blätter v, w strukturell ununterscheidbar, d. h. gibt es einen Automorphismus f mit $f(v) = w$, so ist es bei der Enumeration nicht notwendig an w zwei Blättern anzuhängen, wenn dies bereits bei v geschehen ist. Ob zwei Bäume isomorph sind, kann in linearer Zeit entschieden werden [1].

Enumeration aller Bäume für den $MST(R)$

Mit $\mathcal{MST}_k$ sei die Menge aller ungelabelten Bäume auf k Knoten bezeichnet. Ähnlich wie bei $\mathcal{T}_n$ kann aus $\mathcal{MST}_k$ die Menge $\mathcal{MST}_{k+1}$ gebildet werden, indem für jeden Baum $T \in \mathcal{MST}_k$ und jeden Knoten $v \in V(T)$ ein Blatt angehängt wird und in der dadurch entstandenen Menge isomorphe Bäume entfernt werden. Die Menge $\mathcal{MST}_k$ wird für den minimal spannenden Baum in Distanzgraphen auf Instanzen mit k Terminalen benötigt.[4]

Zusammensetzung aller Komponenten

Im Folgenden wird ein Enumerationsverfahren beschrieben, das alle Topologien $\mathcal{G}_k$ auf k Terminalen erzeugt, die den Lemmata 1–3 genügen. Darunter befindet sich daher auch eine Worstcase-Instanz G^*, und die Lemmata garantieren, dass $\mathcal{G}_k$ endlich ist.

Neben $SMT(R)$ und $RGA(R)$ wird als technisches Hilfmittel außerdem der $MST(R)$ benötigt. Der Grund dafür wird im nächsten Abschnitt ersichtlich werden, wenn eine geeignete Kantengewichtsfunktion zu einer Topologie gefunden werden muss.

Als graphische Darstellung wird das folgende Schema benutzt. Die Terminale werden horizontal angeordnet. Durch horizontale Kanten wird der minimal spannende Baum im Terminaldistanzgraphen dargestellt, wobei die Kanten ebenfalls zur Topologie gehören. Unterhalb der Terminale befindet sich der minimale Steinerbaum, der nach Lemma 3 aus einer vollen Komponente besteht. Oberhalb der Terminale sind die vollen Komponenten der Größe mindestens drei angeordnet, die der Algorithmus wählt.[5] Sofern diese

[4] Für die ersten Werte, s. `http://www.research.att.com/~njas/sequences/A000055`

[5] Auf die genaue Reihenfolge wird im Abschnitt 2.5.3.3 eingegangen.

die Terminalmenge nicht spannen, werden vom Algorithmus noch Komponenten der Größe zwei gewählt, welche Kanten im Terminaldistanzgraphen entsprechen. Welche das genau sind, wird im nächsten Abschnitt diskutiert werden. Beispiele von Graphen, die in diesem Schema dargestellt sind, finden sich in Anhang C.

Formal gehört eine Topologie $[R, OPT, T_{rga}, MST]$ zu $\mathcal{G}_k$, falls gilt:

(i) R ist die Menge der Terminale mit $|R| = k$,

(ii) $OPT \in \mathcal{T}_k$ mit Blättern R repräsentiert den minimalen Steinerbaum,

(iii) $T_{rga} \subseteq \bigcup_{i=3}^{k-1} \mathcal{T}_i$, wobei die Mengen der inneren Knoten von je zwei Bäumen disjunkt, alle Blätter aus R sind und ihre Vereinigung kreisfrei ist,

(iv) $MST \in \mathcal{MST}_k$ und $V(MST) = R$.

Von Hougardy wurde ein Algorithmus implementiert, der alle Topologien aus $\mathcal{G}_k$ enumeriert. Dabei wurden Topologien gelabelt, eine geeignete lexikographische Ordnung definiert und jeweils diejenigen Vertreter ausgegeben, die lexikographisch minimal sind.

Der Vorteil bei diesem Ansatz besteht darin, dass sich die bei der Enumeration auftretende Rekursion abbrechen lässt, wenn ausgehend von dem aktuellen Knoten im Rekursionsbaum nur noch Instanzen erzeugt werden können, die lexikographisch nicht minimal sind.

Die Anzahl an Topologien wächst sehr schnell an, und die ersten Werte sind in der rechten Spalte der Tabelle 2.3 abgebildet. Für $k = 4$ befinden sich alle Topologien in Abbildung 2.8 auf Seite 33 und für $k = 5$ im Anhang C.

k	MST-OPT-Paare	eine Komponente	zwei Komponenten	Gesamt
4	4	9	0	9
5	20	416	247	663
6	228	30061	54472	84533
7	3476	3460561	12593612	$> 16 \cdot 10^6$
8	73389			$> 1.4 \cdot 10^9$

Tabelle 2.3: *Anzahl Topologien auf vier bis acht Terminalen*

2.5.3.3 Zuordnung der Kantengewichte durch ein geeignetes Ungleichungssystem

Im letzten Schritt müssen zu einer gegebenen Topologie die Kanten geeignet gewichtet werden. Erreicht wird dies, indem die Zielfunktion $\frac{RGA(R)}{SMT(R)}$ unter

Einhaltung diverser Nebenbedingungen maximiert wird.[6] Diese Nebenbedingungen sorgen zum einen dafür, dass der Algorithmus die gewünschte Lösung wählt und legen zum anderen den minimalen Steinerbaum fest. Durch die Nebenbedingungen wird der Relative Greedy Algorithmus also quasi simuliert.

Im Folgenden wird das Ungleichungssystem zunächst beschrieben und anschließend erläutert, wie die Zielfunktion auf einen Faktor $1+\epsilon$ für beliebig kleines $\epsilon > 0$ approximiert werden kann.

Notation: Mit $T_1, \ldots, T_{i^*}$ seien die vollen Komponenten der Größe mindestens drei und $T_{i^*+1}, \ldots, T_{i_{max}}$ die vollen Komponenten der Größe zwei bezeichnet, die der Algorithmus als Steinerbaum zurückliefert. Die Menge der Blätter von T_i sei mit t_i bezeichnet.

Ohne Kantengewichte sind bei einer Topologie nur der minimal spannende Baum im Terminaldistanzgraphen, der optimale Steinerbaum und die vom Algorithmus zu wählenden vollen Komponenten der Größe mindestens drei festgelegt. Darüber hinaus gibt es jedoch im Ablauf des Algorithmus einige Freiheitsgrade, die sich in drei Kategorien einordnen lassen:

(i) Die Reihenfolge $\sigma : [i^* \to i^*]$, in der der Algorithmus die Terminalmengen $t_{\sigma(1)}. \ldots, t_{\sigma(i^*)}$ wählt,

(ii) welche Kanten im Distanzgraphen bezüglich einer gewählten Komponente eingespart werden und

(iii) welche Kanten im Distanzgraphen zu der Menge T_i mit $i^*+1 \leq i \leq i_{max}$ gehören, sofern $i^* < i_{max}$.

Mehrdeutigkeiten für die erste Kategorie lassen sich durch Festlegung einer Permutation $\sigma : [i^*] \to [i^*]$ vermeiden. Durch die Ungleichungen

$$\forall t \in R \quad \forall\, j \in [i^*] : f_{j-1}(t_{\sigma(j)}) \leq f_{j-1}(t) \tag{2.18}$$

wird es dem Relativen Greedy Algorithmus dann ermöglicht, die gewünschte Reihenfolge $t_{\sigma(1)}, \ldots, t_{\sigma(i^*)}$ zu wählen.

Desweiteren verschwinden die Mehrdeutigkeiten in der zweiten und dritten Kategorie durch Einführung einer Ordnungsrelation $\preceq$ auf $E(MST)$. Dabei bedeutet $x \preceq y$, dass das Gewicht von y mindestens so groß sein soll wie das von x. Das weitere Vorgehen besteht darin, zu jedem σ und $\preceq$ einer Topologie ein Ungleichungssystem $U_{\sigma,\preceq}(G)$ aufzustellen, dessen Lösung den Kanten diejenigen Gewichte zuweist, die das Verhältnis der Algorithmuslösung zum $SMT(R)$ auf der Topologie unter Berücksichtigung von σ und $\preceq$ maximiert.

[6] Aus Stetigkeits- und Kompaktheitsgründen wird das Maximum angenommen.

Die Relation $\preceq$ gestattet es, die Änderung des minimal spannenden Baumes im Terminaldistanzgraphen bei Wahl einer vollen Komponente zu kontrollieren. Dazu ist das folgende Lemma hilfreich, das einen „umgekehrten" Kruskal-Algorithmus beschreibt und ähnlich bewiesen werden kann.

Lemma 5. *Sei G ein gewichteter zusammenhängender Graph. Weiterhin sei T ein Baum, der durch sukzessives Entfernen der schwersten nichttrennenden Kanten entsteht. Dann ist T minimal spannend in G.*

Ist eine totale Ordnungsrelation $\preceq$ auf $E(MST)$ gegeben, so kann mit diesem Verfahren ein $MST(R/t_1, \ldots, t_i, t)$ bestimmt werden. Durch die Relation $\preceq$ sind die Freiheitsgrade aus der zweiten Kategorie eliminiert. Die Einsparung des minimal spannenden Baumes im Terminaldistanzgraphen bei Wahl der Komponente t sei durch

$$\triangle(R/t_1, \ldots, t_i, t) := \sum_{x \in E(MST(R/t_1,\ldots,t_i))} w(x) - \sum_{x \in E(MST(R/t_1,\ldots,t_i,t))} w(x) \qquad (2.19)$$

notiert.

In jedem Schritt i mit $i \in [i^* + 1, i_{max}]$ ist keine Verbesserung gegenüber dem minimal spannenden Baum im Distanzgraphen mehr möglich, d. h. für alle Komponenten T mit Blättern t gilt $f_i(t) \geq 1$. Gleichheit wird für jede Kante aus $E(MST(R/t_1, \ldots, t_{i^*})) \cap E(MST)$ erzielt, die der Algorithmus in den verbleibenden Schritten wählt, um alle bisher gewählten Komponenten zu verbinden. Damit ist auch die Kantenmenge für die dritte Kategorie bestimmt.

Definition 5. *Sei eine Topologie $G \in \mathcal{G}_k$ gegeben. Eine Gewichtsfunktion $w : E(G) \to \mathbb{R}_+$ respektiert σ und $\preceq$, falls folgende Bedingungen gelten:*

(i) Der Relative Greedy Algorithmus kann die Komponenten in der Reihenfolge $t_{\sigma(1)}, \ldots, t_{\sigma(i^*)}, t_{i^*+1}, \ldots, t_{i^*_{max}}$ wählen,

(ii) $x \preceq y$ impliziert $w(x) \leq w(y)$,

(iii) MST ist ein minimal spannender Baum im Terminaldistanzgraphen,

(iv) die volle Komponente OPT bildet einen minimalen Steinerbaum.

Damit sind jetzt alle Vorüberlegungen abgeschlossen und der Hauptsatz in diesem Abschnitt kann formuliert werden. Sofern G bekannt ist, wird statt $U_{\sigma,\preceq}(G)$ einfach nur $U_{\sigma,\preceq}$ verwendet.

Satz 5. *Zu gegebenen $G \in \mathcal{G}_k$, σ und $\preceq$ existiert genau dann ein Ungleichungssystem $U_{\sigma,\preceq}(G)$, dessen zulässige Lösungsmenge nichtleer ist, wenn eine Gewichtsfunktion $w : E \to \mathbb{R}^0_+$ existiert, die σ und $\preceq$ respektiert.*

Beweis. ($\Rightarrow$) Im Folgenden wird zunächst das Ungleichungssystem $U_{\sigma,\preceq}$ beschrieben, das aus einer zulässigen Lösung eine respektierende Gewichtsfunktion generiert. Es sei daran erinnert, dass die Ungleichungen den Ablauf des Relativen Greedy Algorithmus unter Einhaltung von σ und $\preceq$ simulieren. Dabei sind $w(x)$ mit $x \in E(G)$ gerade die Variablen des Ungleichungssystems. In Abhängigkeit von ihrer Bedeutung lassen sich die Ungleichungen in fünf Gruppen einteilen.

Gruppe 1: Diese Gruppe enthält die Zielfunktion, die Normierung des Optimums, Erfüllung der Ordnung $\preceq$, sowie die Vermeidung negativer Kantengewichte.

Gruppe 2: Die Kanten $E(MST)$ bilden einen minimal spannenden Baum im Terminaldistanzgraphen.

Gruppe 3: Der Relative Greedy Algorithmus wählt im i-ten Schritt die Komponente $t_{\sigma(i)}$.

Gruppe 4: Nach dem i^*-ten Schritt ist gegenüber $MST(R/t_1, \ldots, t_{i^*})$ keine Verbesserung mehr möglich.

Gruppe 5: Die volle Komponente OPT bildet einen minimalen Steinerbaum.

In Abbildung 2.6 auf Seite 28 ist zu einer konkreten Topologie ein Ungleichungssystem abgebildet.

Gruppe 1: Wird die Länge des minimalen Steinerbaums durch

$$\sum_{x \in E(OPT)} w(x) = 1 \tag{2.20}$$

normiert, dann ist die Zielfunktion gerade die Länge des vom Relativen Greedy Algorithmus berechneten Steinerbaums, also

$$\max \sum_{i=1}^{i_{max}} \sum_{x \in E(T_i)} w(x). \tag{2.21}$$

Desweiteren werden die Ungleichungen

$$\forall \{x, y\} \subset E(MST) \text{ mit } x \preceq y : w(x) \leq w(y), \tag{2.22}$$

hinzugefügt. Redundante Ungleichungen, die sich aus der Transitivität von $\preceq$ ergeben, können dabei ausgelassen werden.

Außerdem sollen alle Kantengewichte nichtnegativ sein:

$$\forall x \in E(MST) \cup E(OPT) \cup \bigcup_{i=1}^{i^*} E(T_i) : w(x) \geq 0. \tag{2.23}$$

Die Zielfunktion (2.21) ist gerade das maximale Verhältnis der Algorithmuslösung gegenüber dem minimalen Steinerbaum, dessen Gewicht durch (2.20) mit eins normiert ist. Diese Normierung ist gerechtfertigt aufgrund der Skalierungsinvarianz des Relativen Greedy Algorithmus, d. h. er kann die gleiche Lösung wählen, wenn alle Kantengewichte mit einer positiven Konstanten multipliziert werden. Durch die Normierung wird erreicht, dass die Zielfunktion linear ist.

Gruppe 2: Um zu erzwingen, dass $E(MST)$ minimal spannend ist, ist es nach Lemma 5 hinreichend zu fordern, dass die Länge eines beliebigen xy-Pfades aus der Menge aller xy-Pfade $\mathcal{P}_{xy}$ mindestens so lang ist wie die schwerste Kante auf dem (eindeutigen) xy-Pfad P_{xy}^{MST}, der nur Kanten aus $E(MST)$ benutzt. Dies führt auf die Ungleichungen

$$\begin{aligned} \forall \{x, y\} \subset T: \quad & \forall e \in E(P_{xy}^{MST}) : \forall P_{xy} \in \mathcal{P}_{xy} \\ & w(e) \leq \sum_{e' \in E(P_{xy})} w(e'). \end{aligned} \tag{2.24}$$

Gruppe 3: In dieser Gruppe sind alle Ungleichungen enthalten, die es dem Algorithmus in Schritt i erlauben, die Komponente $t_{\sigma(i)}$ zu wählen. Zum einen muss $t_{\sigma(i)}$ für den Algorithmus mindestens so gut sein, wie eine volle Komponente der Größe zwei, was durch die Ungleichung

$$f_{i-1}(t_{\sigma(i)}) \leq 1 \;\; \text{d. h.} \sum_{x \in E(T_{\sigma(i)})} w(x) \leq \triangle(R/t_{\sigma(1)}, \ldots, t_{\sigma(i-1)}, t_{\sigma(i)}) \tag{2.25}$$

erreicht wird. Desweiteren darf im i-ten Schritt keine volle Komponente T der Größe mindestens drei mit $f_{i-1}(t) < f_{i-1}(t_{\sigma(i)})$ existieren. Dabei genügt es, sich auf volle Komponenten T zu beschränken, die mit $T \cup \bigcup_{j=1}^{i-1} T_j$ keinen Kreis schließen. Die Menge dieser Komponenten sei mit $\mathcal{T}_i'$ bezeichnet und führt dann auf die Ungleichungen

$$\begin{aligned} & \forall\, i \in [i^*] \text{ und } \forall\, T \in \mathcal{T}_i' : \\ & f_{i-1}(t_{\sigma(i)}) \leq f_{i-1}(t) \\ \text{d. h.} \; & \frac{\sum\limits_{x \in E(T_{\sigma(i)})} w(x)}{\triangle(R/t_{\sigma(1)}, \ldots, t_{\sigma(i-1)}, t_{\sigma(i)})} \leq \frac{\sum\limits_{x \in E(T)} w(x)}{\triangle(R/t_{\sigma(1)}, \ldots, t_{\sigma(i-1)}, t)}. \end{aligned} \tag{2.26}$$

Damit ist sichergestellt, dass der Algorithmus im Schritt i die Terminalmenge $t_{\sigma(i)}$ wählen kann.

Gruppe 4: Nachdem der Algorithmus die Terminalmengen $t_{\sigma(1),\dots\sigma(i^*)}$ gewählt hat, soll keine Einsparung gegenüber dem minimal spannenden Baum im Terminaldistanzgraph mehr möglich sein. Analog zu den Ungleichungen von Gruppe 3 muss daher gelten:

$$\begin{aligned} &\forall T \in \mathcal{T}'_{i^*+1} \quad 1 \leq f_{i^*}(t) \\ \text{d. h.} \quad &\triangle(R/t_{\sigma(1)}, \dots, t_{\sigma(i^*)}, t) \leq \sum_{x \in E(T)} w(x). \end{aligned} \tag{2.27}$$

Gruppe 5: Der Steinerbaum OPT soll ein minimaler Steinerbaum sein. Da OPT bereits mit eins normiert wurde, genügen die Ungleichungen

$$\text{für alle Steinerbäume } S\text{: } \sum_{x \in E(S)} w(x) \geq 1. \tag{2.28}$$

Eine zulässige Lösung von $U_{\sigma,\preceq}$ induziert gerade eine Kantenbelegung, die σ und $\preceq$ respektiert. Die Eigenschaft (i) aus Definition 5 wird durch die Gleichungen (2.25) bis (2.27), Eigenschaft (ii) durch (2.22), Eigenschaft (iii) durch (2.24) und Eigenschaft (iv) durch (2.28) sichergestellt.

($\Leftarrow$) Sei w eine Gewichtsfunktion, die σ und $\preceq$ respektiert. Dann gilt dies auch für w' mit $w'(x) := w(x)/smt(R)$. Nach Konstruktion ist w' eine zulässige Lösung von $U_{\sigma,\preceq}$. □

Da der Beweis konstruktiv ist, gilt das folgende Korollar.

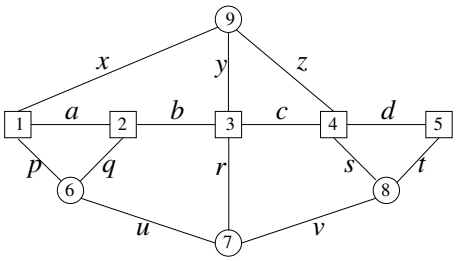
Korollar 2. *Zu einer gegebenen Topologie $G \in \mathcal{G}_k$, σ und $\preceq$ kann $U_{\sigma,\preceq}(G)$ algorithmisch erzeugt werden.*

Ein weiteres Korollar rechtfertigt die Verwendung des Maximums bezüglich der Zielfunktion.

Korollar 3. *Da die Zielfunktion stetig und die Lösungsmenge abgeschlossen in $\mathbb{R}^{|E|}$ ist, wird das Supremum $c(U_{\sigma,\preceq})$ der Zielfunktion von $U_{\sigma,\preceq}$ angenommen, sofern eine zulässige Lösung existiert. $c(U_{\sigma,\preceq}(G))$ ist gerade die maximal mögliche Güte, die eine Kantengewichtsfunktion auf G haben kann, die σ und $\preceq$ respektiert.*

Korollar 4. *Sei $c(U_{\sigma,\preceq}(G))$ der Maximalwert von $(U_{\sigma,\preceq}(G))$ mit $G \in \mathcal{G}_k$. Dann beträgt die Güte des Relativen Greedy Algorithmus auf k Terminalen*

$$\max_{G,\preceq,\sigma} \{c(U_{\sigma,\preceq}(G))\}. \tag{2.29}$$



Gruppe 1:

$$
\begin{aligned}
(1.1): &\quad \max x + y + z + a + d \\
(1.2): &\quad p + q + r + s + \\
&\quad + t + u + v = 1 \\
(1.3): &\quad a \leq b \\
(1.4): &\quad a, b, c, d, x, y, z \geq 0 \\
&\quad p, q, r, s, t, u, v \geq 0
\end{aligned}
$$

Gruppe 3:

$$
\begin{aligned}
12: &\quad a \leq p + q \\
13: &\quad a \leq x + y \\
13: &\quad a \leq p + u + r \\
13: &\quad b \leq x + y \\
13: &\quad b \leq p + u + r \\
14: &\quad a \leq x + z \\
14: &\quad a \leq p + u + v + s \\
14: &\quad b \leq x + z \\
14: &\quad b \leq p + u + v + s \\
14: &\quad c \leq x + z \\
14: &\quad c \leq p + u + v + s \\
15: &\quad a \leq p + u + v + t \\
15: &\quad b \leq p + u + v + t \\
15: &\quad c \leq p + u + v + t \\
15: &\quad d \leq p + u + v + t
\end{aligned}
$$

Gruppe 4:

125: $a + d \leq p + q + t + u + v$
235: $a + d \leq q + r + t + u + v$
245: $a + d \leq q + s + t + u + v$

Gruppe 2:

$$
\begin{aligned}
& \frac{x+y+z}{b+c} \leq 1 \\
12345: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+r+s+t+u+v}{a+b+c+d} \\
1234: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+r+s+u+v}{a+b+c} \\
1235: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+r+t+u+v}{a+b+c} \\
1235: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+r+t+u+v}{a+b+d} \\
1245: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+s+t+u+v}{a+b+d} \\
1245: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+s+t+u+v}{a+c+d} \\
1345: \quad & \frac{x+y+z}{b+c} \leq \frac{p+r+s+t+u+v}{a+c+d} \\
1345: \quad & \frac{x+y+z}{b+c} \leq \frac{p+r+s+t+u+v}{b+c+d} \\
2345: \quad & \frac{x+y+z}{b+c} \leq \frac{q+r+s+t+u+v}{b+c+d} \\
123: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+r+u}{a+b} \\
124: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+s+u+v}{a+b} \\
124: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+s+u+v}{a+c} \\
125: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+t+u+v}{a+b} \\
125: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+t+u+v}{a+c} \\
125: \quad & \frac{x+y+z}{b+c} \leq \frac{p+q+t+u+v}{a+d} \\
134: \quad & \frac{x+y+z}{b+c} \leq \frac{p+r+s+u+v}{a+c} \\
134: \quad & \frac{x+y+z}{b+c} \leq \frac{p+r+s+u+v}{b+c} \\
135: \quad & \frac{x+y+z}{b+c} \leq \frac{p+r+t+u+v}{a+c} \\
135: \quad & \frac{x+y+z}{b+c} \leq \frac{p+r+t+u+v}{a+d} \\
135: \quad & \frac{x+y+z}{b+c} \leq \frac{p+r+t+u+v}{b+c} \\
135: \quad & \frac{x+y+z}{b+c} \leq \frac{p+r+t+u+v}{b+d} \\
145: \quad & \frac{x+y+z}{b+c} \leq \frac{p+s+t+u+v}{a+d} \\
145: \quad & \frac{x+y+z}{b+c} \leq \frac{p+s+t+u+v}{b+d} \\
145: \quad & \frac{x+y+z}{b+c} \leq \frac{p+s+t+u+v}{c+d} \\
234: \quad & \frac{x+y+z}{b+c} \leq \frac{q+r+s+u+v}{b+c} \\
235: \quad & \frac{x+y+z}{b+c} \leq \frac{q+r+t+u+v}{b+c} \\
235: \quad & \frac{x+y+z}{b+c} \leq \frac{q+r+t+u+v}{b+d} \\
245: \quad & \frac{x+y+z}{b+c} \leq \frac{q+s+t+u+v}{b+d} \\
245: \quad & \frac{x+y+z}{b+c} \leq \frac{q+s+t+u+v}{c+d} \\
345: \quad & \frac{x+y+z}{b+c} \leq \frac{r+s+t+v}{c+d}
\end{aligned}
$$

Abbildung 2.6: *Ungleichungssystem der ersten vier Gruppen für eine gegebene Instanz auf fünf Terminalen*

Partielle Ordnungen auf dem MST

Abhängig von der Instanz können Ordnungen $\preceq_1$ und $\preceq_2$ in einem Ungleichungssystem zusammengefasst werden und zwar dann, wenn für alle Algorithmuskomponenten die Einsparung im MST, welche gerade durch den Nenner der linken Seite von (2.26) beschrieben wird, eindeutig ist.

Zur Verdeutlichung diene das Beispiel in Abbildung 2.6. Dort beträgt im ersten Schritt die Einsparung für den MST $\triangle(R/t_1) = c + \max(a, b)$. Dies führt zu der Fallunterscheidung $a \leq b$ und $b \leq a$. Für die folgenden Schritte des Algorithmus können analoge Fallunterscheidungen entstehen. Am Ende müssen statt $(k-1)!$ totalen Ordnungen nur die partiellen Ordnungen berücksichtigt werden, die durch diesen Prozess induziert wurden. Bei der Topologie im Beispiel sind das genau zwei Ordnungen, wobei für $\{1, 2\} \preceq \{2, 3\}$ das Gleichungssystem abgebildet ist. Andererseits führt diese Einsparung an Ordnungen dazu, dass einige zusätzliche Ungleichungen von (2.26) benötigt werden. Für einige konkurrierende Komponenten ist die Einsparung unter Umständen nicht mehr eindeutig. Daher werden alle potenziell möglichen Einsparungen betrachtet und als Ungleichungen hinzugefügt. Dies lässt sich in Abbildung 2.6 anhand der Terminalmengen t mit $t \supset \{3, 5\}$ und $t \not\ni 4$ erkennen. Da keine Aussage getroffen werden kann, ob die Kante c oder d eingespart wird, verdoppelt sich die Anzahl der Ungleichungen für t. Die zusätzlichen Ungleichungen, die nicht die Einsparung im MST beschreiben, werden von dieser dominiert. Daher verringert sich die zulässige Lösungsmenge nicht.

Es ist auch möglich, dass alle Ordnungen simultan behandelt werden können. Wenn in Abbildung 2.6 die Kante $\{1, 9\}$ durch $\{2, 9\}$ ersetzt wird, ist die Einsparung im ersten Schritt gerade $b + c$, so dass keine Fallunterscheidung entsteht.

2.5.3.4 Ein Näherungsverfahren zur Lösung von $U_{\sigma,\preceq}$

Es lässt sich zeigen, dass die Lösungsmenge von $U_{\sigma,\preceq}$ im Allgemeinen nicht konvex ist. Daher lassen sich Verfahren aus der konvexen Optimierung nicht anwenden. Im Folgenden wird ein Näherungsverfahren vorgestellt, um das Ungleichungssystem $U_{\sigma,\preceq}$ zu lösen. Ausgangspunkt für eine approximative Lösung für $U_{\sigma,\preceq}$ ist das folgende Lemma, das die Verwendung linearer Programmierung nahelegt.

Lemma 6. *Sei zu einer Topologie das dazugehörige Ungleichungssystem $U_{\sigma,\preceq}$ gegeben. Zusätzlich seien die Kanten von $MST(R)$ nicht variabel sondern fest. Dann ist $U_{\sigma,\preceq}$ ein lineares Programm.*

Beweis. In $U_{\sigma,\preceq}$ sind die einzigen nichtlinearen Ungleichungen jene in (2.26).

Der Zähler ist linear, und im Nenner befindet sich die Einsparung gegenüber dem minimal spannenden Baum im Distanzgraphen, also eine Summe, bei der jeder Summand die Form $w(x)$ mit $x \in E(MST)$ hat. Werden die Gewichte dieser Kanten festgelegt, wird der Nenner konstant, womit die Ungleichungen in (2.26) linear werden. □

Mit $c(U_{\sigma,\preceq})$ sei der Maximalwert von $U_{\sigma,\preceq}$ benannt, wobei $c(U_{\sigma,\preceq}) := -\infty$ gelte, falls $U_{\sigma,\preceq}$ keine zulässige Lösung hat. Sei $x = (x_1, \dots x_{|R|-1})$ ein Vektor, der als Einträge die Variablen enthält, die den $MST(R)$ repräsentieren. Aus den Ungleichungen (2.20) und (2.23) folgt $0 \leq x_i \leq 1$ für alle $i \in [|R| - 1]$. Die Idee ist nun, diesen $(|R| - 1)$-dimensionalen Einheitswürfel geeignet in eine Menge von Teilwürfeln $\mathcal{W}$ zu unterteilen. Für einen solchen Teilwürfel $W \in \mathcal{W}$ wird zu $U_{\sigma,\preceq}$ zusätzlich die Bedingung $x \in W$ hinzuzufügen. Dieses modifizierte Ungleichungssystem sei mit $U^W_{\sigma,\preceq}$ bezeichnet. Anschließend kann ein geeignetes lineares Programm $LP^W_{\sigma,\preceq}$ formuliert werden, wobei $c(LP^W_{\sigma,\preceq})$ eine obere Schranke für $c(U^W_{\sigma,\preceq})$ ist. Bei zunehmender Verfeinerung von W strebt der Quotient von $c(LP^W_{\sigma,\preceq})$ und $c(U^W_{\sigma,\preceq})$ gegen eins.

Zu einem Würfel W seien $\{x^-, x^+\} \subset W$ die zwei Ecken mit der Eigenschaft

$$\forall (x_1, \dots x_{|R|-1}) \in W : x_i^- \leq x_i \leq x_i^+. \tag{2.30}$$

$LP^W_{\sigma,\preceq}$ ensteht aus $U_{\sigma,\preceq}$, indem (2.30) hinzugenommen wird und in (2.26) jeder Summand x_i im Nenner auf der linken Seite durch x_i^+ und jeder Summand x_j im Nenner auf der rechten Seite durch x_j^- ersetzt wird. Die Ungleichungen in (2.26) werden dadurch abgeschwächt, so dass jede zulässige Lösung in $U^W_{\sigma,\preceq}$ auch zulässig in $LP^W_{\sigma,\preceq}$ ist. Da sich die Zielfunktion nicht geändert hat, gilt $c(U^W_{\sigma,\preceq}) \leq c(LP^W_{\sigma,\preceq})$.

Der Algorithmus zur Approximation von $c(U_{\sigma,\preceq})$ unterteilt W rekursiv in $2^{|R|-1}$ gleich große Teilwürfel, sofern zulässige Lösungen in $LP^W_{\sigma,\preceq}$ existieren und $c(LP^W_{\sigma,\preceq}) > (1 + \epsilon)c'$ gilt. Dabei ist c' der beste bisher gefundene Wert für $U_{\sigma,\preceq}$ und $\epsilon > 0$ eine vorher festgelegte Genauigkeit. c' wird im Laufe des Algorithmus wie folgt aktualisiert: Für jeden Mittelpunkt m eines Teilwürfels gilt $c(U^{\{m\}}_{\sigma,\preceq}) = c(LP^{\{m\}}_{\sigma,\preceq})$ und kann daher berechnet werden. Wenn dieser Wert größer als c' ist, wird c' entsprechend aktualisiert, da eine bessere untere Schranke für $c(U_{\sigma,\preceq})$ gefunden wurde. Der Pseudocode ist in Abbildung 2.7 dargestellt.

Satz 6. *Wenn der Algorithmus aus Abbildung 2.7 zu einem $U_{\sigma,\preceq}$ und $\epsilon > 0$ mit Ausgabe c' terminiert, dann gilt*

$$c(U_{\sigma,\preceq}) \leq (1 + \epsilon) \cdot c'. \tag{2.31}$$

Algorithmus zur Approximation von $c(U_{\sigma,\preceq}(G))$

Input: $U_{\sigma,\preceq}(G)$ mit $G \in \mathcal{G}_k$, $\epsilon > 0$
Output: Eine $(1+\epsilon)$-Approximation für $c(U_{\sigma,\preceq}(G))$
1 $c' = 0$, $x^- = (0,\ldots,0)$, $x^+ = (1,\ldots,1) =: \mathbf{1}$, $(x^-, x^+ \in [0;1]^{k-1})$
2 LP-rek(x^-, x^+)
3 **return** c'

4 LP-rek(x^-, x^+)
5 $m = \frac{x^+ - x^-}{2}$ und sei W der durch x^- und x^+ begrenzte Würfel
6 $c' = \max\{c', c(LP^{\{m\}}_{\sigma,\preceq})\}$
7 **if** $c(LP^W_{\sigma,\preceq}) > (1+\epsilon)\cdot c'$
8 $d = (x_1^+ - x_1^-)/2$
9 **for** $y \in \mathbb{F}_2^{|R|-1}$ **do**
10 LP-rek$(x^- + d\cdot y, x^+ - d\cdot(\mathbf{1}-y))$

Abbildung 2.7: *Algorithmus zur Approximation von $c(U_{\sigma,\preceq})$*

Beweis. Angenommen, es gilt $c(U_{\sigma,\preceq}) > (1+\epsilon)\cdot c'$. Sei x^* ein optimaler Zielvektor für $U_{\sigma,\preceq}$, eingeschränkt auf die Komponenten, die den $MST(R)$ repräsentieren. Betrachte die Iteration mit $x^* \in W$, bei der kein rekursiver Abstieg erfolgte und sei c'' die zu dem Zeitpunkt beste gefundene Lösung von $c(U_{\sigma,\preceq})$. Da kein rekursiver Abstieg erfolgte, galt $c(LP^W_{\sigma,\preceq}) \leq (1+\epsilon)\cdot c''$. Andererseits ist $c(LP^W_{\sigma,\preceq})$ eine obere Schranke für $c(U^W_{\sigma,\preceq})$. Zusammen mit $c'' \leq c'$ und $c(U_{\sigma,\preceq}) = c(U^W_{\sigma,\preceq})$ wegen $x^* \in W$ folgt

$$c(U_{\sigma,\preceq}) = c(U^W_{\sigma,\preceq}) \leq c(LP^W_{\sigma,\preceq}) \leq (1+\epsilon)\cdot c'' \leq (1+\epsilon)\cdot c' \tag{2.32}$$

im Widerspruch zur Annahme. □

Es bleibt der Fall zu betrachten, dass der Algorithmus in eine Endlosrekursion gerät. Dann gibt es eine Folge von Würfeln $\{W_i\}_{i\in\mathbb{N}}$ mit $W_{i+1} \subset W_i$ und für jedes W_i gilt $c(LP^{W_i}_{\sigma,\preceq}) \neq -\infty$. Da das Volumen der Würfel gegen null strebt, gibt es einen Grenzwert $x = \lim_{i\to\infty} W_i$. Sei L_i die Lösungsmenge zu $LP^{W_i}_{\sigma,\preceq}$. Da $L_{i+1} \subseteq L_i$ gilt und jedes L_i nichtleer und abgeschlossen in $\mathbb{R}^{|E|}$ ist, gilt nach einem Satz von Riesz $\cap_{i\in\mathbb{N}} L_i \neq \emptyset$ [58, S. 43]. Damit ist insbesondere die Lösungsmenge von $LP^{\{x\}}_{\sigma,\preceq}$ nichtleer. Der Würfel besteht dann nur aus einem Punkt, und es gilt $LP^{\{x\}}_{\sigma,\preceq} = U^{\{x\}}_{\sigma,\preceq}$. Daher hat auch $U^{\{x\}}_{\sigma,\preceq}$ eine zulässige Lösung. Es kann also nicht passieren, dass der Algorithmus in eine Endlosrekursion gerät, obwohl keine zulässige Lösung für $U_{\sigma,\preceq}$ existiert.

Es ist jedoch möglich, dass c' nicht aktualisiert wird. Dies kann auftreten, wenn bei der Folge $\{W_i\}_{i\in\mathbb{N}}$ der Mittelpunkt von W_i nie eine zulässige Lösung für $U_{\sigma,\preceq}$ ist. Bei allen berechneten Topologien trat dieses Szenario jedoch nie auf und der Algorithmus terminierte stets. Die verwendete Genauigkeit war meistens $\epsilon = 0.01$.

Bemerkung: Anstelle von $LP^W_{\sigma,\preceq}$ kann ein lineares Programm verwendet werden, das eine schärfere obere Schranke für $U^x_{\sigma,\preceq}$ ist. Seien dazu wieder $x_1, \ldots, x_{|R|-1}$ die Variablen, die den $MST(R)$ repräsentieren und X_l und X_r die Indizes der Variablen, die im Nenner von (2.26) nur auf der linken bzw. rechten Seite auftreten. X_b seien die Indizes der Variablen, die auf beiden Seiten im Nenner vorkommen. Ist $i \in X_l$, dann wird wie bisher x_i durch x_i^+, im Falle von $i \in X_r$ durch x_i^- ersetzt. Eine Kurvendiskussion ergibt für $i \in X_b$ dann die Zuweisung:

$$x_i \text{ wird zu } \begin{cases} x_i^+ & \text{falls} \quad \sum_{j\in X_l} x_j^+ < \sum_{j\in X_r} x_j^- \\ x_i^- & \text{falls} \quad \sum_{j\in X_l} x_j^+ > \sum_{j\in X_r} x_j^- \end{cases} . \tag{2.33}$$

Im Falle von $\sum_{j\in X_l} x_j^+ = \sum_{j\in X_r} x_j^-$ kann der gesamte Nenner von (2.26) auf beiden Seiten entfernt werden. Dieses verschärfte lineare Programm wurde auch in der Implementation verwendet und unter Verwendung von CPLEX 7.0 gelöst.

2.5.3.5 Ergebnisse auf Instanzen mit vier bis sechs Terminalen

Nachfolgend werden die Ergebnisse auf Topologien mit vier bis sechs Terminalen diskutiert. Bei drei oder weniger Terminalen findet der Relative Greedy Algorithmus stets das Optimum. Für diesen Abschnitt bedeutet der in kursiv gesetzte Begriff *Güte* einer Topologie $G \in \mathcal{G}_k$, dass auf G eine Kantenbelegung mit $rga(G)/smt(G) = x$ existiert, aber keine mit $rga(G)/smt(G) > 1.01 \cdot x$. Die Fehlertoleranz beträgt daher $\epsilon = 0.01$. Bei den auf den nächsten Seiten abgebildeten Graphen sind die Kantengewichte – ausgehend von den berechneten Werten – nachoptimiert worden. Generell ist eine Clusterbildung zu beobachten, d. h. trotz relativ vieler Topologien treten nur wenige Werte auf. Mit t_1 sei die Terminalmenge bezeichnet, die der Algorithmus im ersten Schritt gewählt hat. Für $|t_1| = 3$ und $|t_1| = 4$ wird im Folgenden auch von Dreier- bzw. Viererkomponenten gesprochen.

4 Terminale

Auf vier Terminalen gibt es 9 Topologien, die in der Abbildung 2.8 dargestellt sind. Bei fünf Topologien ergibt sich eine *Güte* von 7/6 und bei vier Topologien eine von 10/9. Eine *Güte* von 7/6 wird genau dann erreicht, wenn

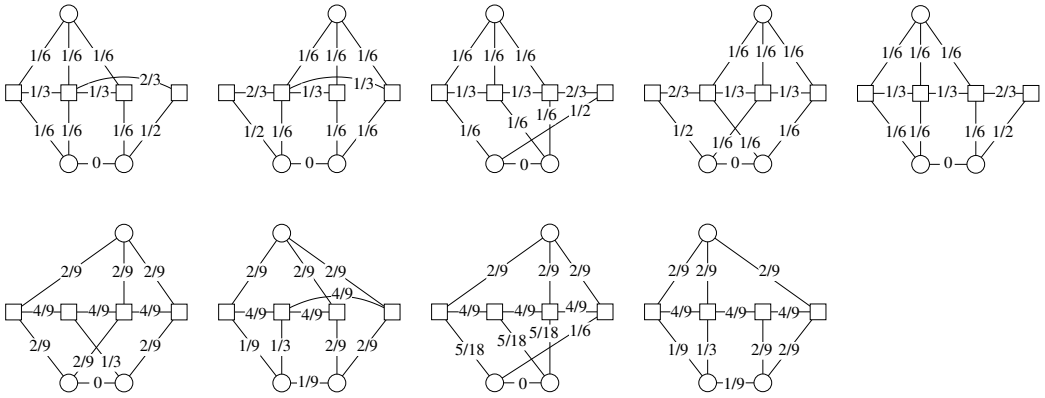


Abbildung 2.8: *Ergebnisse für die neun Topologien mit vier Terminalen. In der oberen Reihe haben die Topologien eine Güte von 7/6, in der unteren von 10/9.*

das Terminal $R \setminus t_1$ ein Blatt im $MST(R)$ ist. Den zu diesem Terminal inzidenten Kanten kann dann ein besonders großes Gewicht gegeben werden.

Es gibt einige weitere Strukturmerkmale. Alle Kanten der Algorithmuskomponente haben das gleiche Gewicht. Bei sieben von neun Topologien hat die Kante, die inzident zu zwei Nichtterminalen ist, das Gewicht null und kann daher kontrahiert werden (vgl. Abbildung 2.4 (a)). Der minimale Steinerbaum ist dann ein Stern. Die Anzahl linearer Programme, die pro MST-Ordnung $\preceq$ für $\epsilon = 0.01$ berechnet werden muss, beträgt mit dem im vorherigen Abschnitt vorgestellten Algorithmus etwa 1000.

5 Terminale

Auf fünf Terminalen gibt es 663 Topologien, die in Anhang C aufgelistet sind. Das Berechnen der *Güte* pro Topologie dauerte bei einer Genauigkeit von $\epsilon = 0.01$ wenige Minuten[7]. Die Anzahl der zu berechnenden linearen Programme betrug bei festem $\preceq$ und σ etwa 90000 und die maximale Rekursionstiefe war neun.

Erstmalig existieren auch Topologien mit zwei Algorithmuskomponenten der Größe drei. Bei den berechneten Werten erzielt die zweite Komponente jedoch keine Einsparung gegenüber $MST(R/t_1)$. Im Folgenden wird daher nicht unterschieden, ob die Instanz aus einer oder zwei Dreierkomponenten besteht.

Die verschiedenen *Güten* sind in der Tabelle 2.4 abgebildet. Eine *Güte* von mindestens 1.193 haben 92 Topologien. Eine dieser Topologien mit skalierten Kantengewichten ist in Abbildung 2.9 dargestellt, wobei nach geeigneter

[7] Alle Angaben beziehen sich auf Computer mit einer Taktfrequenz von 1.8 GHz.

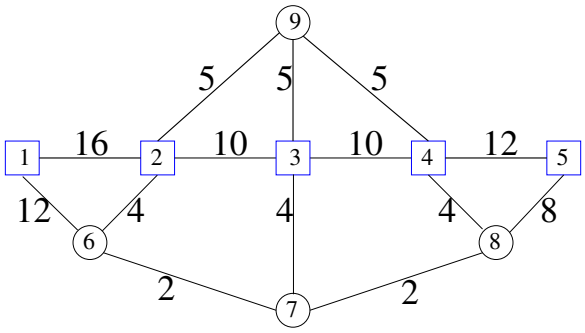


Abbildung 2.9: *Instanz auf fünf Terminalen mit Güte* $1.19\bar{4}$

Kantenoptimierung eine *Güte* von $43/36 = 1.19\bar{4}$ erreicht wird. Generell gilt für diese Klasse: Die rechte Seite jeder Ungleichung aus (2.27) der Gruppe 4 besteht aus fünf Summanden, $|t_1| = 3$ und $\{x, y\} = R \setminus t_1$ sind Blätter im $MST(R)$. Dann gibt es noch einige Topologien mit dieser *Güte*, bei denen x ein Blatt und y dessen Nachbar im $MST(R)$ ist.

Der wesentliche Unterschied bei den Topologien mit *Güte* 1.1911 ist, dass x ein Blatt und y ein innerer Knoten im $MST(R)$ ist.

Alle Topologien mit *Güten* 1.1699 haben dieselbe Eigenschaft, wie bei den 1.1911-Instanzen, nur dass eine Ungleichung der Gruppe 4 existiert, dessen rechte Seite aus vier Summanden besteht. Die restlichen Instanzen mit $|t_1| = 3$ erreichen eine *Güte* von $7/6$, wenn die berechneten Kantenwerte geeignet nachoptimiert werden.

Für $|t_1| = 4$ ist die *Güte* teilweise von $x = R \setminus t_1$ abhängig. Bei allen Instanzen mit Güte 1.1704 oder 1.1693 ist x ein Blatt im $MST(R)$, während x bei allen Instanzen mit Güte 1.125 ein innerer Knoten ist. Bei den Topologien im Bereich 1.1621 bis 1.1655, die ebenfalls auf $7/6$ nachoptimiert werden können, kommen beide Varianten vor.

Generell ist der minimal spannende Baum im Distanzgraphen von untergeordneter Bedeutung. Bei den Topologien, die eine hohe Güte aufweisen, treten alle Spannbäume auf. Bei den Dreierkomponenten sind stets alle Kanten gleichgewichtet, bei einer Viererkomponente ist das Gewicht der Kante zwischen den beiden Nichtterminalen entweder null oder sehr klein gegenüber den anderen Kantengewichten. Die restlichen Kantengewichte einer Viererkomponente sind gleich.

6 Terminale

Auf sechs Terminalen gibt es bereits 84533 Topologien, und die Laufzeit pro Topologie beträgt bei einer Genauigkeit von $\epsilon = 0.01$ bereits einige Stun-

Anzahl Topologien auf 5 Terminalen	untere Schranke für *Güte*
92	1.1930
96	1.1911
21	1.1704
87	1.1699
45	1.1693
198	[1.1621, 1.1655]
124	1.1250

Tabelle 2.4: *Clusterbildung auf fünf Terminalen*

den. Daher konnten nicht mehr alle *Güten* auf sechs Terminalen berechnet werden, und es wurde eine Stichprobe von 1142 Topologien ausgewählt, die ausgehend von den Ergebnissen auf fünf Terminalen gute untere Schranken erwarten lassen. Konkret wurde zunächst eine Stichprobe der Größe 1000 erstellt, indem die drei Bestandteile einer Topologie unabhängig voneinander wie folgt gewählt wurden:

(i) 900 Topologien hatten genau eine Algorithmuskomponente, davon 700 eine Dreier- und 200 eine Viererkomponente. 100 weitere Topologien hatten zwei Algorithmuskomponenten.

(ii) Es waren alle Topologien für den $MST(R)$ vorhanden mit einer leichten Präferenz für Pfade.

(iii) Die beiden Baumtopologien für den minimalen Steinerbaum (vgl. Abbildung 2.5 auf Seite 21) waren gleichhäufig vertreten.

Die Anbindung dieser drei Bestandteile untereinander erfolgte zufällig. Zusätzlich wurden 142 Instanzen zufällig aus den 84533 Topologien ausgewählt.

Als Ergebnis lässt sich wiederum eine Clusterbildung beobachten, wobei die besten gefundenen *Güten* in Tabelle 2.5 abgebildet sind. Wenn nur eine Stichprobe genommen werden kann, ist eine Clusterbildung vorteilhaft, insbesondere wenn das Cluster, das die Güte des Relativen Greedy Algorithmus auf k Terminalen repräsentiert, groß ist. Damit erhöht sich die Wahrscheinlichkeit, auch bei geringer Stichprobengröße eine Worstcase-Instanz zu finden.

Die beste gefundene untere Schranke für die *Güte* beträgt 1.21324, und eine entsprechende Instanz[8] ist in Abbildung 2.10 abgebildet. Anhand der

[8]Bei allen anderen gefundenen Instanzen mit dieser *Güte* besteht die Algorithmuslösung ebenso aus einer vollen Komponente der Größe drei und dann nur noch vollen Komponenten der Größe zwei.

Anzahl	*Güte*	Anzahl	*Güte*
8	1.213240	28	1.205450
2	1.209345	9	1.201602
17	1.207991	1	1.201578
28	1.207616	5	1.201540
11	1.206965	1	1.201015
1	1.206706	7	1.200998
12	1.206465	3	1.200965
1	1.205660	4	1.200348

Tabelle 2.5: *Alle auf der Stichprobe mit sechs Terminalen gefundenen Instanzen mit einer Güte von mindestens 1.2.*

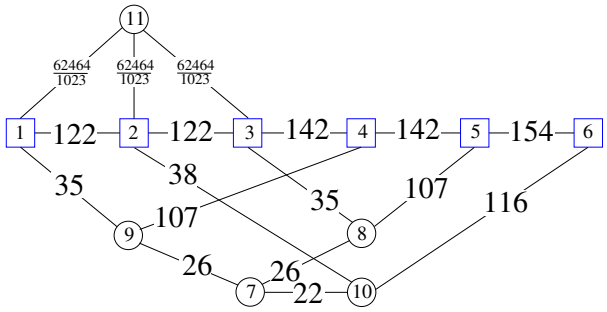


Abbildung 2.10: *Instanz auf sechs Terminalen mit Güte 1.21324*

Kantengewichte lassen sich die zu den Algorithmuskomponenten konkurrierenden Komponenten erkennen. Im ersten Schritt gilt $f_0(\{1,2,3\}) = \frac{62464}{1023}/(2 \cdot 122) = 256/341 = f_0(\{1,2,3,4,5,6\})$. Nach dem ersten Schritt ist keine Einsparung möglich. Der Algorithmus bindet anschließend die Terminale 4, 5 und 6 an die bestehende Teillösung über die MST-Kanten mit Kosten $2 \cdot 142 + 154 = 438$ an. Die Anbindung über die Kantenmenge $\{\{3,8\},\{8,7\},\{7,9\},\{7,10\},\{4,9\},\{5,8\},\{6,10\}\}$ hat Kosten 439 und ist damit teurer.

Generell ist bei diesen Instanzen zu beobachten, dass die Gewichte der Kanten des minimalen Steinerbaums nur sehr wenige Regelmäßigkeiten aufweisen. Daher ist es schwierig, die gefundenen Worstcase-Instanzen derart zu erweitern, um untere Schranken zu konstruieren, die in die Nähe von 1.38538 gelangen oder gar darüber hinaus.

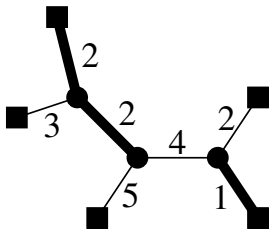


Abbildung 2.11: *Die dicker gezeichneten Kanten bilden den Loss mit Gewicht fünf.*

2.6 Der Loss-Algorithmus

In diesem Abschnitt wird zunächst der Loss-Algorithmus von Robins und Zelikovsky [57] vorgestellt, der zur Zeit der Approximationsalgorithmus für Steinerbäume mit der besten oberen Schranke von 1.550 für die Güte ist. Anschließend wird eine untere Schranke von 1.233 für diesen Algorithmus bewiesen.

2.6.1 Algorithmusbeschreibung

Der Loss-Algorithmus benutzt ähnlich wie der Relative Greedy Algorithmus eine Bewertungsfunktion für eine maximal k-elementige Terminalmenge, wobei k wieder eine zuvor festgelegte Konstante ist. Der wesentliche Unterschied ist, dass beim Loss-Algorithmus im Gegensatz zum Relativen Greedy Algorithmus gewisse Kantenlängen im Steinerbaum auf null gesetzt werden – der sogenannte Loss – und die damit erreichte Einsparung zum minimal spannenden Baum im Terminaldistanzgraphen ins Verhältnis gesetzt wird. Im Folgenden sei $T := SMT(t)$ ein minimaler Steinerbaum auf $t \subset R$ zu einem Graphen G mit Terminalmenge R.

Definition 6. *Sei T mit Terminalmenge t gegeben. Dann ist $LOSS(T)$ ein* kürzester *Teilgraph mit Knotenmenge $V(T)$, in dem jede Zusammenhangskomponente ein Terminal enthält. Die Länge von $LOSS(T)$ wird mit $loss(T)$ bezeichnet.*

Bemerkung: Zu gegebenen T kann der Loss durch eine Variante des Kruskal-Algorithmus in polynomieller Zeit bestimmt werden. Ein Beispiel für einen Loss ist in Abbildung 2.11 dargestellt.

Mit $mst(R/LOSS(t_1, \ldots, t_i))$ sei die Länge eines minimal spannenden Baumes im Terminaldistanzgraphen bezeichnet, nachdem im Graphen G alle Gewichte der Kanten aus $\bigcup_{j=1}^{i} LOSS(T_j)$ auf null gesetzt wurden. Der Al-

gorithmus wählt sich pro Iteration eine Terminalmenge, welche die Funktion

$$f_i(t_{i+1}) := \frac{loss(t_{i+1})}{mst(R/LOSS(t_1,\ldots,t_i)) - mst(R/LOSS(t_1,\ldots,t_{i+1}))} \quad (2.34)$$

minimiert. Außerdem gelte $f_i(t) := 1$ für eine zweielementige Terminalmenge t. Dieses Auswählen von Terminalmengen wird solange iteriert, bis die Vereinigung der korrespondierenden Steinerbäume $T_1,\ldots,T_i$ die Terminalmenge R spannen. Als Länge des gefundenen Steinerbaumes wird $mst(R/LOSS(t_1,\ldots,t_{i_{max}}))+\sum_{i=1}^{i_{max}} loss(t_i)$ zurückgegeben. Der Pseudocode ist in Abbildung 2.12 abgebildet. Ein entsprechender Steinerbaum mit dieser Länge kann anschließend wie folgt konstruiert werden. Nach Zeile 7 werden alle Losskanten in G auf null gesetzt. Danach wird auf diesem Graphen mit der MST-Heuristik ein Steinerbaum T berechnet und in einem Postprocessing jede in T enthaltene Losskante durch ihr ursprüngliches Gewicht ersetzt.

Loss-Algorithmus

Input: $G = (V, E, w)$, $R \subseteq V$
Output: Ein Steinerbaum mit einer Länge maximal $1.550 \cdot smt_k(G)$
1 $i := 0$
2 **while** $T_1,\ldots,T_i$ die Menge R nicht spannen
3 wähle $t_{i+1} \subseteq R, |t_{i+1}| \leq k$, das
4 $f_i(t_{i+1}) := \frac{loss(t_{i+1})}{mst(R/LOSS(t_1,\ldots,t_i)) - mst(R/LOSS(t_1,\ldots,t_{i+1}))}$ minimiert
5 $i := i + 1$
6 $i_{max} := i$
7 **return** $mst(R/LOSS(t_1,\ldots,t_{i_{max}})) + \sum_{i=1}^{i_{max}} loss(t_i)$

Abbildung 2.12: *Loss-Algorithmus*

Robins und Zelikovsky haben den folgenden Satz bewiesen.

Satz 7 ([65])**.** *Seien $G = (V, E)$, $w : E \to \mathbb{R}_+$ ein Graph mit $R \subseteq V$ und $lca_k(G_R)$ die Länge des vom Loss-Algorithmus berechneten Steinerbaums. Dann gilt*

$$lca_k(G_R) \leq \left(1 + \frac{\ln(3)}{2}\right) \cdot smt_k(G_R) \doteq 1.550 \cdot smt_k(G_R). \quad (2.35)$$

Weiterhin haben Robins und Zelikovsky auf der folgenden speziellen Graphenklassen eine deutlich bessere obere Schranke bewiesen.

Definition 7. *Eine Instanz heißt* quasibipartit*, wenn je zwei Nichtterminale paarweise nicht adjazent sind, die Menge der Nichtterminale also eine stabile Menge bildet.*

Satz 8 ([65])**.** *Für quasibipartite Graphen hat der Loss-Algorithmus eine Approximationgüte von maximal* 1.27*.*

Bemerkung: Eine volle Komponente T einer quasibipartiten Instanz ist ein Stern und sein Loss besteht gerade aus einer Kante aus $E(T)$ mit minimalen Gewicht.

Die Einschränkung, auf allgemeinen Instanzen nur k-elementige Terminalmengen zu betrachten, beruht darauf, polynomielle Laufzeit für den Algorithmus zu garantieren. Durch eine geeignete Implementation kann auf quasibipartiten Graphen unter allen $2^{|R|}$ Terminalmengen eine die Auswahlfunktion minimierende Terminalmenge in polynomieller Zeit bestimmt werden. Daher kann auf die Einschränkung, nur k-elementige Terminalmengen betrachten zu dürfen, verzichtet werden. Für quasibipartite Graphen G gilt daher $lca(G_R) \leq 1.27 \cdot smt(G_R)$.

2.6.2 Untere Schranken für den Loss-Algorithmus

Um untere Schranken für den Loss-Algorithmus zu konstruieren, ist das Vorgehen ähnlich wie beim Relativen Greedy Algorithmus. Es werden Instanzen vorgestellt, bei denen jede volle Komponente des minimalen Steinerbaums sich mit jeder vollen Komponente des Algorithmus in maximal zwei Terminalen schneiden. Diese Struktur führt wiederum zu einem Gittergraphen.

Einige Gewichte werden wieder fest und andere variabel gewählt. Die Lösung eines geeigneten Ungleichungssystems weist den Variablen dann Werte zu, die eine gute untere Schranke zur Folge hat.

Satz 9. *Für quasibipartite Graphen hat der Loss-Algorithmus eine Approximationgüte von mindestens* 1.233*.*

Beweis. Betrachte die folgende Gitterinstanz mit k^2+1 Terminalen. Gegeben seien Terminale w_{ij} mit $i, j \in [k]$, ein ausgezeichnetes Terminal w_0, sowie die Nichtterminale $v_1, \ldots, v_k$ und $h_1, \ldots, h_k$. Für alle $1 \leq i \leq k$ sei w_0 mit h_i durch eine Kante mit Gewicht eins verbunden, sowie mit v_i durch eine Kante mit Gewicht a_i. Weiterhin sei jedes w_{ij} mit h_i durch eine Kante mit Gewicht eins verbunden, sowie mit v_i durch eine Kante mit Gewicht b_i. Die Gewichte für a_i und b_i werden später bestimmt, es gelte aber $a_i \leq b_i$. Für $k = 4$ ist der Graph in Abbildung 2.13 dargestellt.

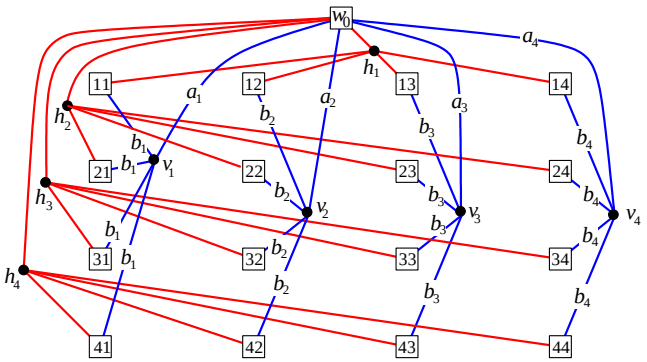


Abbildung 2.13: *Instanz für den Loss-Algorithmus in einem 4×4-Gitter. Die nicht beschrifteten Kanten haben Gewicht eins und die Terminale w_{ij} sind durch ij dargestellt.*

Die Kanten lassen sich disjunkt zerlegen in die Menge, die zu h_i inzident ist (sog. horizontale) Kanten und in die Menge, die zu v_i inzident ist (sog. vertikale Kanten).

Die Kanten des $SMT(R)$ sollen aus den horizontalen Kanten bestehen, deren Kanten alle mit eins gewichtet sind. Daraus ergibt sich eine Länge für den optimalen Steinerbaum von $k(k+1)$. Der Algorithmus soll als Lösung den Steinerbaum bestehend aus den vertikalen Kanten mit einer Länge von $\sum_{i=1}^{k}(a_i + kb_i)$ berechnen. Dazu wird im i-ten Schritt die Kante $\{v_i, w_0\}$ mit Gewicht a_i als Loss ausgewählt, wodurch die Komponenten des minimalen Steinerbaums schrittweise unattraktiver für den Algorithmus werden. Dies erlaubt es, die Kantengewichte a_i und b_i aufsteigend zu gewichten. Nach dem k-ten Schritt soll dann keine Verbesserung mehr möglich sein.

Der minimal spannende Baum im Terminaldistanzgraphen soll durch die (horizontalen) Kanten mit Gewicht eins induziert werden. Damit gilt einerseits $mst(R) = 2k^2$ und führt andererseits zu der Forderung $a_i + b_i \geq 2$ für $1 \leq i \leq k$. Außerdem soll nach dem i-ten Schritt jedes Knotenpaar (w_0, w_{ji}) mit $j \in [k]$ im $MST(R/LOSS(t_1, \ldots, t_i))$ durch eine Kante mit Gewicht b_i verbunden sein, was sich durch die Forderung $b_i < 2$ erreichen lässt.

Im Folgenden wird der i-te Schritt diskutiert, d. h. es wurden bisher die Kanten mit Gewichten $a_1, \ldots a_{i-1}$ als Loss gewählt.

Für eine Terminalmenge $t_{alg} \subseteq N(h(i))$ mit $w_0 \in t_{alg}$ der Größe mindestens drei beträgt die Länge des Loss a_i wegen $a_i \leq b_i$. Im minimal spannenden Baum im Distanzgraphen war für $1 \leq j \leq k$ das Terminal w_{ji} zuvor mit einem Terminal aus $N(v_i) \setminus \{w_{ji}\}$ mit einem Gewicht von zwei verbunden und wird nun durch die Kante $\{w_0, w_{ji}\}$ mit Gewicht b_i ersetzt. Daraus ergibt

sich für die Auswahlfunktion

$$f_{i-1}(t_{alg}) = \frac{a_i}{k(2-b_i)}. \tag{2.36}$$

Ähnlich berechnet sich eine dazu konkurrierende optimale Komponente mit mindestens drei Terminalen t_{opt}. Aus Symmetriegründen sei $t_{opt} \subseteq N(h_1)$, wobei der Loss aus der Kante $\{h_1, x\}$ mit $x \in N(h_1)$ bestehe.

Die Terminale w_{1j} mit $1 \leq j < i$ und $w_{1j'}$ mit $i \leq j' \leq k$ sind jeweils im minimal spannenden Baum im Distanzgraphen mit w_0 durch Kanten der Länge $b_j \in [1;2]$ bzw. je mit einem Terminal aus $N(h_1) \setminus \{w_{1j'}\}$ durch eine Kante der Länge zwei verbunden.

Diese k Kanten werden, sofern t_{opt} gewählt wird, im minimal spannenden Baum im Terminaldistanzgraphen durch k Kanten $\{x, y\}$ mit $y \in N(h_1)\setminus\{x\}$ jeweils der Länge eins ersetzt. Daher ergibt sich

$$f_{i-1}(t_{opt}) = \frac{1}{2(k-i+1) + (\sum_{j=1}^{i-1} b_j) - k}. \tag{2.37}$$

Damit im i-ten Schritt als Loss die Kante mit Gewicht a_i gewählt wird, muss gelten:

$$f_{i-1}(t_{alg}) \leq f_{i-1}(t_{opt}) \iff a_i(k - 2i + 2 + \sum_{j=1}^{i-1} b_j) \leq k(2-b_i). \tag{2.38}$$

Nachdem die Losskanten mit Gewichten $a_1, ..., a_k$ kontrahiert worden sind, soll gegenüber dem minimal spannenden Baum im Distanzgraphen keine Verbesserung mehr möglich sein, d. h. es muss $f_k(t) \geq 1$ für alle Terminalmengen t gelten. Kritisch sind dabei weiterhin horizontale Komponenten t_{opt}. Diese ersetzen im minimal spannenden Baum im Distanzgraphen die Kanten $\{w_0, w_{1j}\}$ mit $1 \leq j \leq k$ je mit Gewicht b_j durch k Kanten je mit Gewicht eins. Damit keine weitere Einsparung erzielt werden kann, muss daher gelten:

$$f_k(t_{opt}) = \frac{1}{\sum_{i=1}^{k} b_i - k} \geq 1. \tag{2.39}$$

Für $k = 4$ ergibt sich bei geeignet gewählten Werten für a_i und b_i bezüglich den Nebenbedingungen $a_i + b_i \geq 2$, $a_i \leq b_i < 2$, (2.38) und (2.39) für die Approximationsgüte des Loss-Algorithmus eine obere Schranke von 1.23389. □

In der Tabelle 2.6 sind die mit Mathematica 6 ermittelten Werte für $2 \leq k \leq 5$. Für $k = 2$ kann Mathematica das Ergebnis exakt berechnen. Diese Instanz ist in [28, Abbildung 10] untersucht worden und die dortigen

	$k=2$	$k=3$	$k=4$	$k=4$ (exakt)	$k=5$
Güte	$\frac{5-\sqrt{2}}{3} \doteq 1.195$	1.22998	**1.23389**	$\frac{211}{140} - \frac{2\sqrt{7/15}}{5}$	1.2
a_1	$2-\sqrt{2}$	0.919916	1	1	1
b_1	$\sqrt{2}$	1.08008	1	1	1
a_2	$2-\sqrt{2}$	0.919916	1.14286	8/7	1.09269
b_2	$3-\sqrt{2}$	1.36217	1.14286	8/7	1.09269
a_3		0.919916	1.26748	$4-4\sqrt{7/15}$	1.22992
b_3		1.55775	1.32099	$-1/7+\sqrt{15/7}$	1.22992
a_4			1.26748	$4-4\sqrt{7/15}$	1.33869
b_4			1.53615	$3-\sqrt{15/7}$	1.33869
a_5					1.33869
b_5					1.33869

Tabelle 2.6: *Untere Schranken für den Loss-Algorithmus*

Kantengewichte stimmen mit den hier berechneten exakten Werten überein. Für $k = 4$ wird das Maximum angenommen, und unter der Annahme $x_2 = y_2 = 8/7$ können auch exakte Werte berechnet werden. Ab $k \geq 5$ nimmt die Güte ab und strebt für $k \to \infty$ gegen eins. Im Anhang A befinden sich für $k = 4$ die Ungleichungen als Mathematica-Skript.

Gu Ji-Xing und Zheng Shi-Bao haben eine Instanz mit einer Güte von $515/420 \approx 1.226$ konstruiert [38]. Diese ist eine Erweiterung der Instanz aus [28, Abbildung 11] und ist neben der schlechteren Güte deutlich komplizierter und auch nicht quasibipartit.

Ein Spezialfall von quasibipartiten Graphen sind die Graphen, bei denen alle Kanten, die zum selben Nichtterminal adjazent sind, gleiches Gewicht haben. Diese Graphenklasse wird uniform-quasibipartit genannt. Im folgenden Abschnitt wird der MSS-Algorithmus vorgestellt, dessen Approximationsgüte auf uniform-quasibipartiten Instanzen 73/60 beträgt [28]. Aber bereits jetzt kann das folgende Korollar festgehalten werden.

Korollar 5. *Der MSS-Algorithmus hat für uniform-quasibipartite Instanzen eine bessere Approximationsgüte als der Loss-Algorithmus.*

Beweis. Die Approximationsgüte des MSS-Algorithmus beträgt $73/60 \leq 1.217$. Wird bei dem obigen Graphen statt $a_i \leq b_i$ Gleichheit gefordert, ergibt sich für $a_1 = 1$, $a_2 = 8/7$, $a_3 = 56/43$ und $a_4 = 2408/1639$ eine Güte von $\frac{2424451}{1973356} \approx 1.2286$. Ein entsprechendes Gleichungssystem befindet sich im Anhang A. □

2.7 MSS-Algorithmus

In [4] ist der MSS-Algorithmus[9] vorgestellt worden, der in einem Hypergraphen, bei dem jede Kante zu maximal k Knoten inzident ist, die Länge eines minimal spannenden Subgraphen approximiert. Die folgende Reduktion zeigt, dass das k-Steinerbaumproblem ein Spezialfall für einen minimal spannenden Subgraphen in einem Hypergraphen ist. Seien ein gewichteter Graph $G = (V, E, w), R \subseteq V$ und ein gewichteter Hypergraph $H = (R, E', w')$ gegeben. Zu genau jeder in G vollen Komponente mit Terminalmenge t und Größe maximal k sei in H die Hyperkante t mit $w'(t) := smt(t)$ vorhanden. Diese Reduktion ist für konstantes k in polynomieller Zeit berechenbar, und ein minimaler k-Steinerbaum in G entspricht gerade einem minimal spannenden Subgraphen in H. Der MSS-Algorithmus kann daher auch verwendet werden, um Steinerbäume in Graphen zu berechnen.

Interessanterweise besitzt der MSS-Algorithmus eine Approximationsgüte von $73/60 = 1.21\bar{6}$ auf der Klasse der *uniform*-quasibipartiten Graphen [28]. Uniform-quasibipartite Graphen sind quasibipartite Graphen, bei denen alle Kanten, die zum selben Nichtterminal adjazent sind, gleiches Gewicht haben. Diese spezielle Graphenklasse tritt bei den besten bekannten Reduktionen für die Nichtapproximierbarkeit des allgemeinen Steinerbaumproblems auf [13; 60] und ist daher auch von theoretischem Interesse.

Der MSS-Algorithmus passt in das bisherige Schema der vorgestellten Algorithmen, die eine Auswahlfunktion benutzen. Auf dem leeren Graphen mit $|R|$ Knoten startend werden solange Hyperkanten aus H eingefügt, bis diese R spannen. Sind bereits die Kanten $t_1, \ldots, t_{i-1}$ gewählt worden, so wird eine Kante t eingefügt, welche die Funktion

$$f_{i-1}(t) = \frac{w'(t)}{c_{i-1}(t)} \tag{2.40}$$

minimiert. Dabei ist $c_{i-1}(t)$ die Differenz der Anzahl an Zusammenhangskomponenten im Hypergraphen $(R, \{t_1, \ldots, t_{i-1}, t\})$ und $(R, \{t_1, \ldots, t_{i-1}\})$.

Für quasibipartite Graphen kann die Forderung nach einem konstanten k entfallen. Der dazugehörige Hypergraph kann zwar exponentiell viele Kanten haben, aber in jedem Schritt kann eine Hyperkante, die die Auswahlfunktion minimiert, in polynomieller Zeit bestimmt werden.

[9] MSS steht für minumum spanning subset.

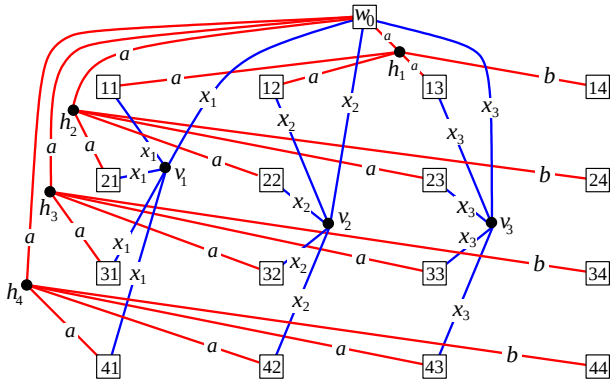


Abbildung 2.14: *Instanz für den MSS-Algorithmus*

2.7.1 Der MSS-Algorithmus auf quasibipartiten Instanzen

Im Gegensatz zu uniform-quasibipartiten Graphen ist keine oberen Schranken für die Approximationsgüte für quasibipartite Graphen bekannt. Im Folgenden wird eine untere Schranke von $1.30\bar{5}$ für den MSS-Algorithmus auf quasibipartiten Graphen gezeigt. Die Instanz ist dabei eine Abwandlung aus der unteren Schranken für den Loss-Algorithmus.

Satz 10. *Für quasibipartite Graphen beträgt die Approximationsgüte des MSS-Algorithmus mindestens* $\frac{47}{36}$.

Beweis. Betrachte die Gitterinstanzen vom Loss-Algorithmus (s. Seite 39) ohne den Knoten v_k und die dazu inzidenten Kanten und der folgenden veränderten Gewichtsfunktion. Alle Kanten, die zu dem Knoten v_i für $i \in [k-1]$ inzident sind, haben das Gewicht x_i. Alle Kanten, die zu dem Terminal w_{jk} für $j \in [k]$ inzident sind, haben das Gewicht b. Alle restlichen Kanten haben das Gewicht a. Für $k = 4$ ist der Graph in Abbildung 2.14 abgebildet.

Der minimale Steinerbaum bestehe aus den k Sternen mit Mittelpunkt h_i und Gewicht $k(ka + b)$. Die vom MSS-Algorithmus berechnete Steinerbaum bestehe aus den $k-1$ Sternen mit Mittelpunkt v_i, sowie den Kanten $\{w_{i4}, h_i\}$ und $\{w_{i1}, h_i\}$ für jedes $1 \leq i \leq k$. Die Kanten werden im folgenden so gewählt, dass der Algorithmus im i-ten Schritt den Stern mit Mittelpunkt v_i wählen kann und anschließend jeweils das Terminal w_{jk} über die Kanten $\{w_{jk}, h_j\}$ und $\{h_j, w_{j1}\}$ für jedes $1 \leq j \leq k$ anbinden kann.

Ähnlich wie beim Relativen Greedy Algorithmus lässt sich auch hier ein Ungleichungssystem aufstellen. Die optimale Lösung sei normiert, d. h. es gelte $k(ka + b) = 1$. Die Zielfunktion ist mit $(k-1)\sum_{i=1}^{k-1} x_i + k(a+b)$ gerade die Länge der Algorithmuslösung. Weiterhin soll $x_i \leq x_j$ für $i < j$

	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
Güte	$\frac{5}{4}=1.25$	$\frac{13}{10}=1.3$	$\frac{47}{36}=1.30\bar{5}$	$\frac{109}{84}\approx 1.298$	$\frac{37}{28}\approx 1.285$
a	1/4	1/15	1/24	1/35	1/48
b	1/8	2/15	1/12	2/35	1/24
x_1	1/6	3/40	2/45	5/168	3/130
x_2	—	1/10	1/20	2/63	5/224
x_3	—	—	1/15	1/28	1/42
x_4	—	—	—	1/21	3/112
x_5	—	—	—	—	1/28

Tabelle 2.7: *Untere Schranken für den MSS-Algorithmus*

gelten, sowie alle Kantengewichte nichtnegativ sein. Für den Algorithmus gibt es dann im i-ten Schritt zwei konkurrierende volle Komponenten mit den Terminalmengen $\{w_{ji},\ldots,w_{jk}\}$ und $\{w_{ji},\ldots,w_{(j-1)k}\}$. Es muss daher für alle $1 \leq i < k$ gelten:

$$\frac{k+1}{k}\cdot x_i \leq \min\left(\frac{(k+1-i)a+b}{k+1-i}, \frac{(k+1-i)a}{k-i}\right). \tag{2.41}$$

Wenn die letzte Ungleichung in zwei einzelne Ungleichungen zerlegt wird, ergibt sich für konstantes k ein lineares Programm, das für $k=4$ im Anhang B zu finden ist. In Tabelle 2.7 sind für $k \leq 2 \leq k = 6$ die optimalen Kantenbelegungen abgebildet. Eine untere Schranke von 47/36 für den MSS-Algorithmus in quasibipartiten Graphen ergibt sich (ebenfalls wie beim Loss-Algorithmus) gerade für $k=4$. □

Korollar 6. *Der MSS-Algorithmus hat für quasibipartite Instanzen eine schlechtere Approximationsgüte als der Loss-Algorithmus.*

2.7.2 Der MSS-Algorithmus auf allgemeinen Instanzen

Sei eine Topologie gegeben, bei der die optimale Lösung und die Reihenfolge σ der vom MSS-Algorithmus gewählten vollen Komponenten festgelegt sind. Wie beim Relativen Greedy Algorithmus kann dann ein Gleichungssystem aufgestellt werden, dessen Lösung eine Worstcase-Instanz bezüglich σ induziert. Dabei sind alle Ungleichungen sowie die Zielfunktion linear, sofern die optimale Lösung mit eins normiert.

Satz 11. *Die Approximationsgüte für den MSS-Algorithmus auf allgemeinen Steinerbäumen beträgt mindestens* 1.5.

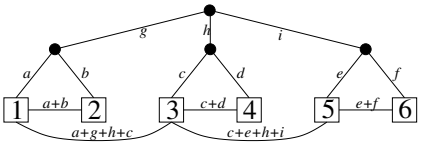


Abbildung 2.15: *Instanz mit einer Güte von 1.5 für den MSS-Algorithmus auf nicht-quasibipartiten Instanzen*

Beweis. Betrachte die Instanz in Abbildung 2.15 mit $a = b = 8$, $c = d = 10$, $e = f = 12$, $g = 7$, $h = 5$ und $i = 18$.

Der minimale Steinerbaum besteht dann aus einer vollen Komponente mit Gewicht 84. Der MSS-Algorithmus kann hingegen nacheinander die Kanten $\{1, 2\}$, $\{3, 4\}$, $\{5, 6\}$, $\{1, 3\}$ und $\{3, 5\}$ wählen; genaueres zum dazugehörigen linearen Programm befindet sich in Anhang B. Die Güte des MSS-Algorithmus beträgt daher mindestens $(2a+b+3c+d+2e+f+g+2h+i)/84 = 1.5$. Das dazugehörige lineare Programm ist im Anhang B zu finden. □

2.8 Ungewichtete quasibipartite Graphen

Ein Spezialfall der quasibipartiten Graphen sind *ungewichtete* quasibipartite Graphen. Für diese Graphenklasse ist das Berechnen eines minimalen Steinerbaums weiterhin $\mathcal{APX}$-schwer, und der Algorithmus mit der besten bisher bekannten Approximationsgüte von maximal $8/7 - 1/160 \doteq 1.137$ basiert auf einem Matroid-Paritäts-Algorithmus [5].

Bei diesen Instanzen hängt die Länge eines Steinerbaums direkt mit der Anzahl k an verwendeten vollen Komponenten der Größe mindestens drei zusammen, da dieser aus $|R| + k$ Knoten besteht und dessen Länge daher $|R| + k - 1$ ist. Ein optimaler Steinerbaum verwendet daher eine minimale Anzahl an notwendigen vollen Komponenten der Größe mindestens drei.

Die Restriktion nur k-elementige Terminalmengen beim Relativen Greedy Algorithmus und Loss-Algorithmus zu betrachten besteht darin, in polynomieller Zeit eine Menge zu finden, welche die Auswahlfunktion minimiert. Im Fall von uniform-quasibipartiten Graphen kann diese Restriktion entfallen, da in polynomieller Zeit die die Auswahlfunktion minimierende volle Komponente gefunden werden kann. Im Folgenden seien diese beiden Algorithmen entsprechend modifiziert.

Im Folgenden sei die Einschränkung gefordert, dass keine Kante e zu je zwei Terminalen inzident ist. Kommt e nicht in $SMT(R)$ vor, so kann e hinzugefügt werden und eine andere Kante auf dem entstehenden Kreis entfernt werden ohne das Gewicht zu ändern. In einem Preprocessing können daher alle diese Kanten kontrahiert werden.

Ausgehend von dieser Einschränkung wird in diesem Abschnitt gezeigt, dass der MSS-Algorithmus, der Relative Greedy Algorithmus und der Loss-Algorithmus für diese Graphenklasse zusammenfallen. Da die Approximationsgüte des MSS-Algorithmus auch auf dieser Graphenklasse 73/60 beträgt [29], haben der Loss-Algorithmus und der Relative Greedy Algorithmus daher ebenfalls eine Güte von 73/60.

Für die Äquivalenz vom Relativen Greedy Algorithmus und dem MSS-Algorithmus genügt es zu zeigen, dass im i-ten Schritt eine Terminalmenge t genau dann die Auswahlfunktion für den Relativen Greedy Algorithmus minimiert, wenn t – aufgefasst als Hyperkante – auch die Auswahlfunktion für den MSS-Algorithmus minimiert.

Lemma 7. *Seien $t_1, \ldots, t_{i-1}$ die bisher vom Relativen Greedy Algorithmus gewählten Terminalmengen und die vom MSS-Algorithmus gewählten Hyperkanten. Wenn der Relative Greedy Algorithmus im i-ten Schritt t und der MSS-Algorithmus t' wählt, dann gilt $f_i^{rga}(t) = f_i^{rga}(t')$ und $f_i^{mss}(t') = f_i^{mss}(t)$.*

Beweis. Im $MST(R/t_1, \ldots, t_{i-1})$ gibt es nur Kanten mit Gewicht zwei und null. Zwei Terminale x und y sind dort genau dann durch einen Pfad ausschließlich mit Nullkanten verbunden, wenn sie im Hypergraphen $(R, \{e(t_1), \ldots, e(t_{i-1})\})$ in der selben Zusammenhangskomponente liegen. Für $x \in t$ gilt $y \notin t$ wegen $f_i^{rga}(t \backslash \{x\}) < f_i^{rga}(t)$, analoges gilt für t'. Bei Wahl von t beträgt die Einsparung im Terminaldistanzgraphen daher $2(|t| - 1)$. Analog beträgt die Differenz der Anzahl an Zusammenhangskomponenten bei Einfügung von t' im Hypergraphen $|t'| - 1$. Für die Auswahlfunktionen ergibt sich daher $f_i^{rga}(t) = |t|/(2(|t| - 1))$ bzw. $f_i^{mss}(t') = |t'|/(|t'| - 1)$, was auf die Ungleichungskette

$$\begin{aligned} f_i^{rga}(t) \leq f_i^{rga}(t') &= \frac{1}{2} \cdot f_i^{mss}(t') \\ &\leq \frac{1}{2} \cdot f_i^{mss}(t) = \frac{1}{2} \cdot 2 \cdot f_i^{rga}(t) = f_i^{rga}(t). \end{aligned} \tag{2.42}$$

führt. Es gilt daher $f_i^{rga}(t) = f_i^{rga}(t')$ und analog $f_i^{mss}(t) = f_i^{mss}(t')$. □

Korollar 7. *Durch Induktion über die Anzahl Schritte folgt, dass beide Algorithmen auf ungewichteten quasibipartiten Instanzen, bei denen keine Kante zu zwei Terminalen inzident ist, äquivalent sind.*

Die Äquivalenz zwischen dem Loss-Algorithmus und dem Relativen Greedy Algorithmus kann ähnlich gezeigt werden. Im Distanzgraphen $MST(R/LOSS(t_1, \ldots, t_{i-1}))$ gibt es nur Kanten mit Gewicht eins und zwei. Eine Kante e hat im Distanzgraphen das Gewicht eins, wenn ein Nichtterminal v im Ausgangsgraphen mit $Loss(t_j) = \{x, v\}$ und $x \in e$ für ein $j \in [i-1]$

existiert. Hauptaussage des folgenden Lemmas ist, dass beide Algorithmen in jeder Iteration eine Terminalmenge nehmen können, die zum gleichen Nichtterminal benachbart ist. Der Effekt ist, dass im Distanzgraphen vom Loss-Algorithmus zwei Knoten durch einen Pfad jeweils mit Kantengewicht eins genau dann verbunden sind, wenn sie durch einen Pfad der Länge null im Distanzgraphen für den Relativen Greedy Algorithmus verbunden sind. Einsparungen im Distanzgraphen für den Relativen Greedy Algorithmus bzw. Loss-Algorithmus können daher nur entstehen, wenn Kanten mit Gewicht zwei durch Kanten mit Gewicht eins bzw. null ersetzt werden.

Lemma 8. *Seien $t_1, \ldots, t_{i-1}$ die vom Relativen Greedy und $t'_1, \ldots, t'_{i-1}$ die vom Loss-Algorithmus gewählten Komponenten. Weiterhin sei $LOSS(t'_j) = \{x_j, y_j\}$ mit $y_j \in V \setminus T$ und es gelte $t_j \subseteq N(y_j)$ für $1 \leq j < i$. Dann gilt:*

(i) Minimiert t die Funktion f_i^{rga}, so minimiert t auch f_i^{loss}.

(ii) Minimiert t' die Funktion f_i^{loss} mit $LOSS(t') = \{x, y\}$ und $y \in V \setminus T$, so existiert $t \subseteq N(y)$, das f_i^{rga} minimiert.

Beweis. Zu (i): Angenommen, der Relative Greedy Algorithmus wählt im i-ten Schritt t. Dann gilt $f_i^{rga}(t) = |t|/(2(|t|-1))$. Weiterhin beträgt die Einsparung $|t|-1$ im Distanzgraphen vom Loss-Algorithmus, es gilt daher $f_i^{loss}(t) = 1/(|t|-1)$. Angenommen, es gibt eine Menge t' mit $f_i^{loss}(t') = 1/d < 1/(|t|-1)$ und y sei das inzidente Nichtterminal zur Losskante. Dann gibt es eine $d+1$ elementige Teilmenge $t^* \subset N(y)$ mit $f_i^{rga}(t^*) = (d+1)/(2d)$. Da $(d+1)/(2d)$ streng monoton fallend und $d > |t|-1$ ist, folgt

$$f^{rga}(t^*) = \frac{d+1}{2d} < \frac{|t|-1+1}{2(|t|-1)} = f_i^{rga}(t), \tag{2.43}$$

was ein Widerspruch zur Wahl von t ist.

Zu (ii): Angenommen, der Loss-Algorithmus wählt t' mit $f_i^{1oss}(t') = 1/d$. Für eine geeignete Teilmenge $t \subseteq N(y)$ mit $|t| = d+1$ gilt dann $f_i^{rga}(t) = (d+1)/(2d)$. Angenommen, es gibt eine Menge t^* mit $f_i^{rga}(t^*) = |t^*|/(2(|t^*|-1)) < (d+1)/(2d)$. Da $(d+1)/(2d)$ streng monoton fallend ist, gilt $|t^*| > d+1$ und

$$f_i^{loss}(t^*) = \frac{1}{|t^*|-1} < \frac{1}{d} = f_i^{loss}(t') \tag{2.44}$$

im Widerspruch zur Wahl von t'. □

Korollar 8. *Angenommen, der Loss-Algorithmus und der Relative Greedy Algorithmus haben in den Schritten $j \in [i-1]$ jeweils die Blätter eines Sternes mit dem gleichen Mittelpunkt $m_j \in V \setminus R$ gewählt. Wenn der*

Loss-Algorithmus im i-ten Schritt die Blätter eines Sternes mit Mittelpunkt $m_i \in V \setminus R$ wählt, genau dann kann der Relative Greedy Algorithmus auch die Blätter eines Sternes mit Mittelpunkt m_i wählen.

Damit haben der Relative Greedy Algorithmus und der Loss-Algorithmus auf ungewichteten quasibipartiten, bei denen nur Kanten zwischen Terminalen und Nichtterminalen existieren, die gleiche Güte.

2.9 Zusammenfassung und Ausblick

Im ersten Teil dieser Arbeit wurden einige Approximationsalgorithmen für Steinerbäume vorgestellt und untere Schranken bewiesen.

Zum Schluss dieses ersten Teils sind in der Tabelle 2.8 die Ergebnisse über untere und obere Schranken für die vorgestellten Graphenklassen und Algorithmen noch einmal zusammengefasst.

Eine nach wie vor offene Frage ist die Approximationsgüte der vorgestellten Algorithmen. Zwar konnten einige untere Schranken verbessert werden, der Abstand zwischen oberer und unterer Schranke bleibt aber groß, und im Endeffekt ist es eine Glaubensfrage, welche Schranke man eher als „Wahrheit“ ansieht oder ob sie irgendwo in der Mitte vermutet wird.

Ein übliches Vorgehen ist es daher, spezielle Klassen von Graphen zu betrachten – hier waren es quasibipartite, uniform-quasibipartite und ungewichteten quasibipartite Graphen – und dort bessere Approximationsgüten nachweisen zu können, in der Hoffnung, daraus auch für den allgemeinen Fall Nutzen ziehen zu können. Exemplarisch soll das Vorgehen am Loss-Algorithmus dargelegt werden. Aus der Analyse für die obere Schranke geht hervor, dass quasibipartite Instanzen, bei denen der optimale Steinerbaum aus mehreren vollen Komponenten der Größe sechs besteht, keine geeigneten Kandidaten für gute untere Schranken sind. Es wäre interessant zu wissen, ob eine vergleichbare Aussage auch für den allgemeinen Fall gilt. Jedenfalls ist keine Instanzenfamilie der Art wie die in 2.5.2.1 vorgestellte bekannt, bei der der Maximalwert für die Güte der Instanzenfamilie nicht für eine endliche Anzahl an Terminalen angenommen wird. Desweiteren sind die besten bekannten unteren Schranken für den allgemeinen Fall quasibipartit. Es ist daher möglich, dass gerade quasibipartite Instanzen auch Worstcase-Instanzen für den Loss-Algorithmus im allgemeinen Fall darstellen. In dem Fall würde sich die obere Schranke von 1.55 auf 1.27 verbessern. Andererseits resultieren die besten Schranken für die Nichtapproximierbarkeit des Steinerbaumproblems gerade aus quasibipartiten Instanzen. Für diese ist aber eine deutlich bessere obere Schranke bezüglich ihrer Approximierbarkeit bekannt.

quasibipartite Instanzen		Rel. Greedy Algorithmus	Loss-Algorithmus	MSS-Algorithmus
ungewichtet	untere Schranke	73/60	73/60	73/60
	obere Schranke	73/60	73/60	73/60
quasi uniform	untere Schranke		1.22859	73/60
	obere Schranke		1.27	73/60
gewichtet	untere Schranke		1.23389	47/36=1.305
	obere Schranke		1.27	
allgemein				
	untere Schranke	1.38	1.23389	1.5
	obere Schranke	1.69	1.550	

Tabelle 2.8: *Übersicht über die Approximationsgüten der Algorithmen*

Bei der Enumeration der Klasse $\mathcal{G}_k$ stellt sich die Frage, ob sie durch weitere Struktureigenschaften eingeschränkt werden kann. Bei den betrachteten Topologien war der $MST(R)$ von untergeordneter Bedeutung, eine naheliegende Frage ist z. B., ob stets eine Worstcase-Instanz auf k Terminalen existiert, bei der der $MST(R)$ ein Stern ist. Aufgrund seiner großen Automorphismengruppe würde die Enumeration erheblich einschränkt werden und darüber hinaus weitere Einblicke in den Algorithmus verschaffen. Weiterhin ist offen, ob und wie das Ungleichungssystem $U_{\sigma,\preceq}$ exakt gelöst werden kann.

Teil II

Konstruktion von Bicliquen in dichten Graphen

Kapitel 3

Einleitung

Im zweiten Teil dieser Arbeit wird ein Algorithmus vorgestellt, der in dichten Graphen einen vollständigen bipartiten Subgraph gewisser Mindestgröße in polynomieller Zeit konstruiert. Als Motivation dienen dabei sogenannte Bicluster, die informal einen bipartiten Subgraphen mit einer deutlich größeren Dichte als der Ausgangsgraph darstellen und Anwendungen im Bereich des maschinellen Lernens, im Data Mining und bei Genexpressionsdaten aus der Molekularbiologie haben [12; 61; 52]. Ein typisches Anwendungsbeispiel bei Genexpressionsdaten ist etwa das Folgende. Gegeben seien eine Menge von Genen $\{1, \ldots, n\}$ und eine Menge von Bedingungen $\{1, \ldots, m\}$, wobei (i, j) gerade das Verhalten von Gen i unter der Bedingung j beschreibt. Gesucht wird nun eine Teilmenge G von Genen und eine Teilmenge B von Bedingungen, so dass die Gene aus G unter den Bedingungen B ein „ähnliches“ Verhaltensmuster aufweisen. Die durch G und B induzierte Submatrix wird ein *Bicluster* genannt. Die Qualität von Biclustern ist dabei sehr abhängig von der konkreten Bewertungsfunktion, die insbesondere durch die jeweiligen Anwendungen geprägt wird. Daher existieren eine Vielzahl an Biclusterdefinitionen und Heuristiken, um Bicluster zu finden. Einen Überblick bietet die Arbeit [49]. Auch von theoretischer Seite zeigt sich, dass die Definition eines „guten“ Biclusters schwierig ist. Ähnlich wie bei normalen Clusterverfahren [46] zeigt sich auch bei Biclustern, dass eine konsistente Biclusterdefinition nicht existiert, sofern bereits wenige, aber durchaus wünschenswerte, Eigenschaften an ein Bicluster gefordert werden [51]. Bei den meisten in der Praxis verwendeten Biclusterdefinitionen zeigt sich dann, dass das Finden eines „besten“ Biclusters $\mathcal{NP}$-schwer ist. Als Algorithmen werden meist Greedystrategien, lokale Suchverfahren, Spektraltechniken oder eine Kombination davon verwendet, wobei Güteaussagen nur in sehr wenigen Fällen bekannt sind [56].

In dieser Arbeit werden die Anforderungen an ein Bicluster verschärft,

indem nicht nur dichte bipartite Subgraphen sondern vollständige bipartite Subgraphen darunter zu verstehen sind. In Anlehnung an die Begriffe Bicluster und Clique werden daher vollständige bipartite Subgraphen im Folgenden kurz *Bicliquen* genannt. In der Darstellung einer Adjazenzmatrix A eines Graphen bilden eine Teilmenge A_Z der Zeilen und eine Teilmenge A_S der Spalten eine Biclique genau dann, wenn die dazugehörige Submatrix nur aus Einsen besteht.

Verwandt mit dem Finden einer „größten“ Biclique ist die Überdeckung eines Graphen mit möglichst wenigen Bicliquen. Eine Bicliquenüberdeckung kann zur Graphkompression genutzt werden, welche wiederum die Laufzeit einiger Algorithmen auf dichten Graphen verbessert [23; 62].

In Abhängigkeit von dem verwendeten Maß für die Größe einer Biclique existieren verschiedene Varianten, wobei bei nichtbipartiten Graphen zusätzlich unterschieden werden kann, ob die Biclique induziert sein soll oder nicht. Die am häufigsten vorkommenden Varianten werden kurz vorgestellt, einen Überblick über weitere $\mathcal{NP}$-vollständige Varianten finden sich in [15].

Bei der **knotenmaximalen Variante** ist eine Biclique gesucht, die die Anzahl Knoten maximiert. In bipartiten Graphen nimmt diese Variante eine Sonderstellung ein, da im Gegensatz zu den anderen Varianten polynomielle Algorithmen bekannt sind. Werden in einem bipartiten Graphen zwischen den Partitionen alle Kanten durch Nichtkanten ersetzt und umgekehrt, so transformiert sich eine größte stabile Menge in eine größte Biclique und umgekehrt. Da eine größte stabile Menge in einem bipartiten Graphen mit dem Algorithmus von Hopcroft und Karp in $\mathcal{O}(\sqrt{|V|}|E|)$ berechnet werden kann [35], ist auch das Berechnen einer knotenmaximalen Biclique in einem bipartiten Graphen in $\mathcal{P}$. Auf einem zufälligen bipartiten Graphen mit Kantenwahrscheinlichkeit 1/2 zwischen den Partitionen besteht eine größte Biclique mit hoher Wahrscheinlichkeit gerade aus der größeren Partition. Für nichtbipartite Graphen ist die knotenmaximale Variante hingegen $\mathcal{NP}$-vollständig [33].

Analog zur knotenmaximalen Variante gibt es die **kantenmaximale Variante**. Diese ist $\mathcal{NP}$-vollständig, was zunächst für den gewichteten Fall gezeigt [15] und später von Peeters auf den ungewichteten Fall verallgemeinert wurde [53].

Auf einem zufälligen Graphen mit konstanter Kantenwahrscheinlichkeit p beträgt die Anzahl Kanten einer größten Biclique mit hoher Wahrscheinlichkeit etwa $i \cdot p^i|V|$ mit $\frac{i-1}{i} \leq p < \frac{i}{i+1}$.

Hochbaum hat einen Approximationsfaktor von zwei nachgewiesen, wenn als Maß die Anzahl zu löschender Kanten gezählt wird, um eine kantenmaximale Biclique zu erhalten [33]. Weiterhin wird in der Arbeit gezeigt, dass

ein Approximationsfaktor von $2-\epsilon$ für ein beliebiges $\epsilon \in (0,1)$ dazu genutzt werden kann, um Vertex-Cover auf einen Faktor $2-\epsilon'$ für ein geeignetes $\epsilon' > 0$ zu approximieren. Unter der Voraussetzung, dass Vertex-Cover nicht auf einen Faktor $2-\epsilon^*$ für jedes $\epsilon^* > 0$ approximiert werden kann [44], ist das Ergebnis von Hochbaum daher bestmöglich.

In der **balancierten Variante** wird eine Biclique mit maximaler Knotenzahl[1] gesucht, bei der beide Partitionen gleiche Kardinalität haben. Sowohl bei der kanten- als auch bei der knotenmaximalen Variante zeigt bei zufälligen Graphen die größte Biclique ein unausgeglichenes Verhältnis hinsichtlich ihrer Partitionsgrößen. Dies wird bei der balancierten Variante vermieden. $\mathcal{NP}$-Vollständigkeitsbeweise finden sich bereits in [26, GT24] und [39]. Im Folgenden sei die Größe einer balancierten Biclique mit Partitionen V_1 und V_2 gerade $|V_1|$. Im zweiten Teil dieser Arbeit liegt der Schwerpunkt auf balancierte Bicliquen, und es wird für dichte Graphen erstmalig ein Algorithmus vorgestellt, der in polynomieller Zeit eine Biclique der Größe $\Omega(\sqrt{\log(|V|)})$ konstruiert.

Teil II ist wie folgt aufgebaut. Im weiteren Verlauf dieses Kapitels wird auf die Verwandtschaft zum Cliquenproblem eingegangen sowie die für den zweiten Teil benötigten Notationen eingeführt. Im darauf folgenden Kapitel wird zunächst skizziert, dass dichte bipartite Graphen eine balancierte Biclique der Größe $\Omega(\log(n))$ besitzen. Anschließend wird ein polynomieller Algorithmus vorgestellt, der in dichten Graphen eine balancierte Biclique der Größe $\Omega(\sqrt{\log(n)})$ berechnet.

3.1 Die Verbindung zum Cliquenproblem

Von ihrer Formulierung ähneln sich das Cliquen- und Bicliquenproblem. Es ist daher nicht so überraschend, dass die ersten Härteresulate aus einer Reduktion von Clique stammen. Die verwendeten Reduktionen sind alle nicht approximationserhaltend, und daher lassen sich daraus keine Resultate bezüglich der Nichtapproximierbarkeit vom Bicliquenproblem ableiten. Von Clique ist bekannt, dass unter der Voraussetzung $\mathcal{P} \neq \mathcal{NP}$ eine größte Clique in polynomieller Zeit nicht auf einen Faktor $n^{1-\epsilon}$ für jedes $\epsilon > 0$ approximiert werden kann [32; 66].

Auf der anderen Seite ist eine bessere Approximationsgüte für das balancierte bzw. kantenmaximale Bicliquenproblem nicht bekannt. Es ist daher durchaus denkbar, dass für Biclique und Clique die selben Nichtapproximierbarkeitsresultate gelten. Feige und Kogan haben die Vermutung aufgestellt,

[1] Äquivalent dazu ist es, die Anzahl Kanten zu maximieren.

dass sowohl die kantenmaximale als auch die balancierte Biclique sich nicht auf einen Faktor n^ϵ für ein geeignetes $\epsilon > 0$ in polynomieller Zeit approximieren lassen [24]. Immerhin zeigen die beiden folgenden Resultate, dass eine deutlich bessere Approximationsgüte unwahrscheinlich ist.

Satz 12 (Feige, Kogan [24])**.** *Wenn das balancierte Bicliquenproblem in polynomieller Zeit für jedes $\delta > 0$ auf einen Faktor $2^{\log(n)^\delta}$ approximiert werden kann, dann ist $3 - SAT \in DTIME(2^{n^{3/4+\epsilon}})$ für jedes $\epsilon > 0$. Wenn die kantenmaximale Variante in polynomieller Zeit auf einen konstanten Faktor approximiert werden kann, dann kann auch MAX-Clique auf einen Faktor von $n/2^{c\sqrt{\log(n)}}$ für ein geeignetes $c > 0$ in polynomieller Zeit approximiert werden.*

Desweiteren haben die Autoren in der gleichen Arbeit gezeigt, dass das balancierte Bicliquenproblem – wie das Cliquenproblem – eine selbstverbessernde Reduktion besitzt. Sofern es sich also in polynomieller Zeit auf einen konstanten Faktor approximieren lässt, besitzt es auch ein polynomielles Approximationsschema.

3.2 Notationen und Definitionen

In diesem Abschnitt werden die für den zweiten Teil notwendigen Notationen und Definitionen kurz vorgestellt.

Zunächst wird die Notation für die Nachbarschaft aus dem ersten Teil angepasst.

Definition 8. *Sei ein Graph $G = (V, E)$ mit $x \in V$ und $W \subseteq V$ gegeben. Dann bezeichnet $N_W(x) := N(x) \cap W$ die auf W eingeschränkte Nachbarschaft von x.*

Die folgenden Definitionen sind Standardbegriffe aus der Graphentheorie.

Definition 9. *Sei ein Graph $G = (V, E)$ gegeben. Dann ist $H = (V', E')$ ein* schwacher Subgraph *von G, falls $V' \subseteq V$ und $E' \subseteq E \cap \binom{V'}{2}$ gilt. H heißt induzierter Subgraph, falls $E' = E \cap \binom{V'}{2}$ gilt.*

Definition 10. *Ein Graph $G = (V, E)$ heißt* bipartit*, wenn V in zwei disjunkte Mengen X und Y zerlegt werden kann, so dass für jede Kante $\{x, y\} \in E$ gilt: $x \in X$ und $y \in Y$. Im Folgenden wird ein bipartiter Graph auch durch $(X \cup Y, E)$ notiert.*

Definition 11. *Sei ein Graph $G = (V, E)$ gegeben. Dann ist $[V_1 \cup V_2]$ eine* Biclique*, falls $V_1, V_2 \subseteq V$ mit $V_1 \cap V_2 = \emptyset$ und für alle $v_1 \in V_1$, $v_2 \in V_2$ die Kante $\{v_1, v_2\}$ in G existiert. Bei einer* balancierten Biclique *wird außerdem*

$|V_1| = |V_2|$ *gefordert und die* Größe *einer balancierten Biclique durch* $|V_1|$ *bezeichnet.*

In bipartiten Graphen entfällt die Unterscheidung zwischen schwachen und stark induzierten Bicliquen.

Definition 12. *Sei ein bipartiter Graph* $G = (X \cup Y, E)$ *gegeben. Dann ist seine Kantendichte* α *definiert durch*

$$\alpha = \frac{|E|}{|X| \cdot |Y|}. \tag{3.1}$$

Weiterhin bezeichne $e(X', Y')$ *für Teilmengen* $X' \subseteq X$ *und* $Y' \subseteq Y$ *die Anzahl Kanten zwischen* X' *und* Y'*. Mit* $\alpha(X', Y') := \frac{e(X',Y')}{|X'| \cdot |Y'|}$ *sei die lokale Dichte auf* X' *und* Y' *bezeichnet.*

Offensichtlich liegt die Dichte eines Graphen stets im Intervall $[0, 1]$. Allgemein wird für nichtbipartite Graphen (V, E) die Dichte durch $|E|/\binom{|V|}{2}$ definiert.

Kapitel 4

Konstruktion von balancierten Bicliquen in dichten Graphen

In diesem Kapitel wird zunächst die Bicliquenkonstruktion auf bipartite Graphen eingeschränkt. Anschließend wird ein kurzer Abstecher in das Gebiet der extremalen Graphentheorie gemacht und die Existenz von balancierten Bicliquen logarithmischer Größe in dichten Graphen skizziert. Als Hauptresultat wird ein polynomieller Algorithmus vorgestellt, der in dichten Graphen eine Biclique der Größenordnung $\Omega(\sqrt{\log(n)})$ konstruiert.

4.1 Einschränkung auf bipartite Graphen

Im Folgenden wird das Bicliquenproblem nur auf bipartiten Graphen betrachtet, was durch die folgende Reduktion motiviert wird. Betrachte zu einem gegebenen Graphen $G = (\{v_1, \ldots, v_n\}, E)$ den bipartiten Graphen $G' = (\{v_1, \ldots, v_n\} \cup \{v'_1, \ldots, v'_n\}, E')$ mit

$$E' = \{\{v_i, v'_j\} \mid \{v_i, v_j\} \in E\}. \tag{4.1}$$

Es lässt sich leicht einsehen, dass in der kantenmaximalen und balancierten Variante eine (schwache) Biclique $[\{v_{i_1}, \ldots, v_{i_k}\} \cup \{v_{j_1}, \ldots, v_{j_l}\}]$ in G eine Biclique $[\{v_{i_1}, \ldots, v_{i_k}\} \cup \{v'_{j_1}, \ldots, v'_{j_l}\}]$ in G' induziert und umgekehrt.

4.2 Das Problem von Zarankiewicz

Es lässt sich zeigen, dass für jeden endlichen Graphen G ein n_0 und ein $\alpha < 1$ existiert, so dass jeder Graph mit Dichte α und mindestens n_0 Kanten G als (schwachen) Subgraphen enthält. Dies führt auf die folgende Definition.

Definition 13. *Sei $ex_n(G)$ die minimale Anzahl an Kanten, so dass jeder Graph mit n Knoten und mindestens dieser Anzahl an Kanten den Graphen G als (schwachen) Subgraphen enthält. Analog sei $ex_{n,n}(G')$ die minimale Anzahl an Kanten, so dass jeder bipartite Graph mit Partitionsgröße je n und mindestens dieser Anzahl an Kanten den bipartiten Graphen G' als (schwachen) Subgraphen enthält.*

Die genaue Bestimmung von $ex_n(G)$ ist eine klassische Frage in der extremalen Graphentheorie. Eine in ihrer Struktur durchaus überraschende Aussage liefert der folgende Satz.[1]

Satz 13 (Erdős, Simonovits [21], 1966)**.** *Sei $\chi(G)$ die chromatische Zahl von G. Dann gilt*

$$ex_n(G) = \left(1 - \frac{1}{\chi(G) - 1}\right) \cdot \binom{n}{2} + o(n^2). \tag{4.2}$$

Der Spezialfall für $ex_n(K_r(t))$ wurde bereits 1946 von Erdős und Stone beantwortet [22], wobei $K_r(t)$ ein vollständig r-partiter Graph ist und jede Partition genau t Knoten enthält.

Für bipartite Graphen ($r = 2$ bzw. $\chi(G) = 2$) ergibt sich nur die Aussage, dass bereits bei einer hinreichend großen aber subquadratischen Kantenanzahl sich balancierte Bicliquen konstanter Größe nicht vermeiden lassen. Der genaue Wert von $ex_{n,n}(K_{k,k})$ ist auch als „Problem von Zarankiewicz“ bekannt und eine bessere obere Schranke als Satz 13 liefert der folgende Satz.

Satz 14 (Kövari, Sós, Turán [47], 1954)**.** *Für alle $k, n \in \mathbb{N}$ gilt*

$$ex_{n,n}(K_{k,k}) \leq (k-1)^{1/k} \cdot n^{2-1/k} + (k-1) \cdot n. \tag{4.3}$$

Beweisskizze (Details siehe z. B. [40])

Es genügt zu zeigen, dass für einen bipartite Graphen $G = (X \cup Y, E)$ mit $n = |X| = |Y|$ ohne eine balancierte Biclique der Größe k die Ungleichung $|E| \leq (k-1)^{1/k} \cdot n^{2-1/k} + (k-1) \cdot n$ gilt. Die Hauptidee des Beweises besteht in einem doppelten Zählargument. Sei dazu $S(x,k) := |\{Z : Z \subseteq N(x) \wedge |Z| = k\}|$ die Anzahl Sterne mit k Kanten und Mittelpunkt x in G.

Es gilt dann $S(x,k) = \binom{|N(x)|}{k}$ und mit Hilfe der Jensenschen Ungleichung folgt

$$\sum_{x \in X} S(x,k) = \sum_{x \in X} \binom{|N(x)|}{k} \geq n \cdot \binom{(\sum_{x \in X} |N(x)|)/n}{k} = n \cdot \binom{|E|/n}{k}. \tag{4.4}$$

[1] Hier wird nur ein Spezialfall betrachtet; der Satz gilt allgemeiner auch für endliche Graphenfamilien.

Auf der anderen Seite kann es für eine feste Menge $Y' \subseteq Y$ mit $|Y'| = k$ maximal $k-1$ Knoten in X geben, die alle Y' als Nachbarschaft enthalten, da ansonsten G eine balancierte Biclique der Größe k hat. Daher gilt $\sum_{x \in X} S(x,k) \leq (k-1)\binom{n}{k}$ und mit (4.4) folgt $n \cdot \binom{|E|/n}{k} \leq (k-1)\binom{n}{k}$. Nach einigen Standardabschätzungen für die Binomialkoeffizenten folgt dann die Behauptung. ⌞⌝

Auf der anderen Seite lässt sich für $ex_{n,n}(K_{k,k})$ auch eine untere Schranke von $n^{2-2/k}$ angeben. Die Methode basiert auf der probabilistischen Methode, wie sie 1959 von Erdős und Réyni [20] eingeführt wurde. Betrachtet wird ein zufälliger bipartiter Graph $G_{n,n,p(n)}$ mit Partitionsgröße je n, bei dem jede Kante mit Wahrscheinlichkeit $p(n)$ vorhanden ist, unabhängig von den anderen Kanten. Dann ist die erwartete Anzahl an Kanten $n^2 p(n)$ und $\binom{n}{k}^2 p(n)^{k^2}$ der Erwartungswert für die Anzahl balancierter Bicliquen der Größe k. Bei einem geeignet gewählten $p(n)$ gibt es in Relation zur Kantenanzahl nur wenige Bicliquen der Größe k. In einem zweiten Schritt wird jede $k \times k$-Biclique durch Entfernen einzelner Kanten zerstört, was bei geeignetem $p(n)$ die Existenz von Graphen mit mehr als $n^{2-2/k}$ Kanten ohne eine Biclique der Größe k zur Folge hat. Genauere Ausführungen finden sich in [19] und eine formale Definition von Zufallsgraphen z. B. in [8].

Für den Beweis von Satz 14 wird die Voraussetzung, dass k eine Konstante ist, nicht verwendet. Der Faktor $(k-1)^{1/k}$ aus (4.3) strebt für $k \to \infty$ gegen 1. Mit dem Ansatz $n^{2-1/k} = \alpha n^2 \iff k = \log_{1/\alpha}(n)$ ergibt sich, dass jeder bipartite Graph mit Dichte α eine balancierte Biclique der Größe $(1-o(1))\log_{1/\alpha}(n)$ besitzt. Ein analoger Ansatz für die untere Schranke von $ex_{n,n}(K_{k,k})$ zeigt die Existenz von bipartiten Graphen mit Dichte α ohne eine Biclique der Größe $2\log_{1/\alpha}(n)$.

Aus algorithmischer Sicht ist diese Situation unbefriedigend. Seit langer Zeit ist bekannt, dass jeder bipartite Graph mit quadratisch vielen Kanten eine balancierte Biclique logarithmischer Größe besitzt, aber es ist nicht bekannt, wie diese in polynomieller Zeit konstruiert werden kann. Generell ist das Phänomen, dass sich die Existenz von Objekten mit probabilistischen Methoden nachweisen lassen, aber ihre explizite Konstruktion offen bleibt, nicht unbekannt. Ein Beispiel dafür sind z. B. Expander [34].

4.3 Bicliquenkonstruktion der Größe $\Omega(\sqrt{\log(n)})$

Im vorherigen Abschnitt wurde skizziert, dass jeder bipartite Graph mit quadratisch vielen Kanten eine balancierte Biclique logarithmischer Größe besitzt. In diesem Abschnitt wird nun ein Algorithmus vorgestellt, der auf dich-

ten Graphen eine balancierten Biclique der Größe $O(\sqrt{\log(n)})$ konstruiert. Genauer wird zu jeder Dichte $0 < \alpha < 1$ die Existenz eines $n_0 = n_0(\alpha)$ gezeigt, so dass der Algorithmus für jeden bipartiten Graphen $G = (X \cup Y, E)$ mit $n = |X| = |Y| \geq n_0$ und Dichte α eine balancierte Biclique der Größe $\Omega(\sqrt{\log(n)})$ in polynomieller Zeit konstruiert.

In den folgenden Abschnitten wird der Algorithmus vorgestellt, die Korrektheit bewiesen und auf die Laufzeit eingegangen.

4.3.1 Balancierter Bicliquen-Algorithmus

Die Grundidee des folgenden Algorithmus besteht darin, in einem bipartiten Graphen $(X \cup Y, E)$ eine bereits gefundene unbalancierte Biclique $[X', Y']$ mit $X' \subset X$, $Y' = \bigcap_{x' \in X'} N(x')$ und $|X'| < |Y'|$ sukzessive um ein geeignetes $x \in X \setminus X'$ zu erweitern, so dass eine neue Biclique $[X' \cup \{x\}, Y' \cap N(x)]$ entsteht. Bei diesem Ansatz wächst also die Anzahl der zur Biclique gehörenden Knoten der Partition X, während die zur Partition Y gehörenden Knoten in der Biclique abnimmt. Dieser Algorithmus wird dabei solange iteriert, bis zum ersten mal eine Biclique entsteht, bei der die meisten Knoten zur Partition X gehören. Da einmal zur Biclique gewählte Knoten aus X in der Biclique verbleiben, gehört der Algorithmus zur Klasse der Greedyalgorithmen.

Zur genaueren Formulierung des Algorithmus werden Folgen von Mengen X_i, Y_i und x_i mit den Eigenschaften konstruiert:

- $X_0 = X$, $Y_0 = Y$
- $X_{i+1} \subset X_i$, $x_{i+1} \in X_i \setminus X_{i+1}$
- $Y_{i+1} = Y_i \cap N(x_{i+1})$

Nach Konstruktion bildet $[\{x_1, \ldots, x_i\}, Y_i]$ eine Biclique. Dabei beschreibt von der Semantik die Menge X_i in der $i+1$-ten Iteration die potentielle Knotenmenge, um die in X enthaltene aktuelle Bicliquenpartition zu erweitern. Mit Hilfe von i'

$$i' := \max_i \{i : i \leq |Y_i| \wedge |X_i| > 1\} \tag{4.5}$$

kann dann eine balancierte Biclique $[\{x_1, \ldots, x_{i'}\}, Y']$ mit $Y' \subseteq Y_{i'}$ und $|Y'| = i'$ konstruiert werden. Damit ist i' die Größe der balancierten Biclique und gleichzeitig die Anzahl Iterationen im Algorithmus.

Im Folgenden wird erläutert, wie in der $i+1$-ten Iteration der Knoten x_{i+1} ausgewählt wird. Dazu werden die Mengen

$$X_i^{(j)} := \{x \in X_i : 2^{j-1}\alpha_i|Y_i| \leq |N_{Y_i}(x)| < 2^j\alpha_i|Y_i|\} \tag{4.6}$$

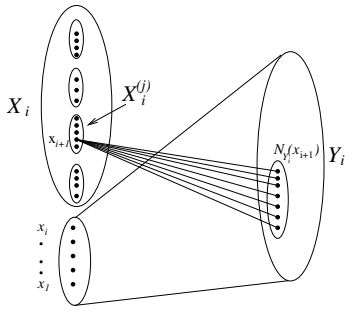


Abbildung 4.1: *Schematische Darstellung eines Iterationsschrittes.* $\tilde{X} := \{x_1, \ldots, x_i\}$ *ist vollständig mit* Y_i *verbunden. Für die nächste Iteration gilt* $X_{i+1} := X_i^{(j)} \setminus \{x_{i+1}\}$, $Y_{i+1} := N_{Y_i}(x_{i+1})$ *und* $\tilde{X}$ *wird um* x_{i+1} *erweitert.*

mit $0 \leq j \leq \lceil \log_2(1/\alpha_i) \rceil$ gebildet, wobei $\alpha_i := \alpha_i(X_i, Y_i)$ die lokale Dichte innerhalb der Partitionen X_i und Y_i ist. Die Knoten in X_i mit einem Grad von mindestens $\alpha_i|Y_i|/2$ – also mindestens dem halben Durchschnittsgrad von X_i – werden derart in Partitionen zusammengefasst, so dass der Minimal- und Maximalgrad innerhalb einer Partition sich höchstens um einen Faktor zwei unterscheiden. Die Knoten in X_i mit einem Grad kleiner als $\alpha_i|Y_i|/2$ stehen dabei für x_{i+1} nicht zur Verfügung und deren Anzahl kann durch Lemma 10 nach oben abgeschätzt werden.

Es wird jetzt ein j gewählt, das $|X_i^{(j)}|$ maximiert. Lemma 11 zeigt dann die Existenz, eines $x_{i+1} \in X_i^{(j)}$ mit $e((X_i^{(j)}), N_{Y_i}(x_i)) \geq \frac{\alpha_i}{8}|N_{Y_i}(x_i)| \cdot |X_i^{(j)}|$. In die aktuelle Biclique wird dann x_{i+1} hinzugefügt und die Mengen $X_{i+1} := X_i^{(j)} \setminus \{x_{i+1}\}$, $Y_{i+1} = Y_i \cap N(x_{i+1})$ gebildet. Abbildung 4.1 veranschaulicht einen Iterationsschritt, und in Abbildung 4.2 ist der Pseudocode vom Algorithmus angegeben.

Wesentlich für die mögliche Anzahl an Iterationen ist die Kontrolle über die Größen $|X_i|$, $|Y_i|$ und $\alpha(X_i, Y_i)$. Einerseits ist es wünschenswert, dass $|X_i|$ und $|Y_i|$ relativ groß sind. Andererseits führt ein zu kleines $\alpha(X_i, Y_i)$ dazu, dass eine hinreichend häufige Iteration der While-Schleife in Abbildung 4.2 nicht gewährleistet werden kann. Desweiteren ist nur eine hohe Dichte $\alpha(X_i, Y_i)$ auch nicht zwangsläufig hilfreich. Besteht X_i aus sehr wenigen Knoten, dann sind ebenfalls nicht mehr hinreichend viele Iterationen möglich. Die Einteilung von X_i in Partitionen $X_i^{(j)}$ bietet daher einen Mittelweg, um alle drei Größen kontrollieren zu können. Der Hauptteil der folgenden Analyse besteht darin zu zeigen, dass $|X_i|$, $|Y_i|$ und $\alpha(X_i, Y_i)$ nicht zu schnell abnehmen, um eine Mindestanzahl an Iterationen garantieren zu können.

Balancierter Bicliquenalgorithmus

Input: $G = (X \cup Y, E)$ mit $|X| = |Y|$
Output: Eine balancierte Biclique
1 $i := 0$, $X_0 := X$, $Y_0 := Y$, $\alpha_0 := \frac{|E|}{|X||Y|}$
2 while $i < |Y_i|$ und $|X_i| > 1$
3 wähle $j \in \mathbb{N} \cup \{0\}$, das $|X_i^{(j)}|$ mit
4 $X_i^{(j)} := \{x \in X_i : 2^{j-1}\alpha_i|Y_i| \leq |N_{Y_i}(x)| < 2^j\alpha_i|Y_i|\}$ maximiert
5 wähle $x_{i+1} \in X_i^{(j)}$ mit $e((X_i^{(j)}), N_{Y_i}(x_i)) \geq \frac{\alpha_i}{8}|N_{Y_i}(x_i)| \cdot |X_i^{(j)}|$
6 $X_{i+1} := X_i^{(j)} \setminus \{x_{i+1}\}$, $Y_{i+1} := N_{Y_i}(x_{i+1})$, $\alpha_{i+1} = \alpha(X_{i+1}, Y_{i+1})$, $i = i+1$
7 $i_{max} := i - 1$
8 return $[\bigcup_{i=1}^{i_{max}} \{x_i\}, Y']$ mit $|Y'| = i_{max}$ und $Y' \subseteq Y_{i_{max}}$

Abbildung 4.2: *Algorithmus für eine balancierte Biclique*

4.3.2 Analyse des Algorithmus

Für den Beweis sind einige Vorbereitungen notwendig. Das folgende Lemma wird sich für die Abschätzung der verbleibenden Dichte im Restgraphen als nützlich erweisen.

Lemma 9. *Sei ein bipartiter Graph $G = (X \cup Y, E)$ mit Kantendichte α gegeben. Dann gilt*

$$\sum_{y \in Y} |N(y)|^2 \geq |Y| \left(\alpha |X|\right)^2 . \tag{4.7}$$

Beweis. Sei $Y = \{y_1, \ldots, y_{|Y|}\}$ gegeben. Unter Verwendung der Cauchy-Schwarzschen-Ungleichung

$$< a, b >^2 \leq ||a||^2 \cdot ||b||^2 \tag{4.8}$$

ergibt sich für den normierten Vektor a mit Einträgen $a_i = 1/\sqrt{|Y|}$ für

$i \in [|Y|]$ und b mit Einträgen $b_i = |N(y_i)|$ die Abschätzung

$$\begin{aligned}
\sum_{y \in Y} |N(y)|^2 &= \left(\sum_{i=1}^{|Y|} a_i^2 \right) \cdot \left(\sum_{i=1}^{|Y|} b_i^2 \right) \\
&\geq \left(\sum_{i=1}^{|Y|} a_i b_i \right)^2 \\
&= \left(\sum_{i=1}^{|Y|} \frac{1}{\sqrt{|Y|}} \cdot |N(y_i)| \right)^2 \\
&= \frac{1}{|Y|} \left(\sum_{i=1}^{|Y|} |N(y_i)| \right)^2 \\
&= \frac{1}{|Y|} \left(\alpha |X||Y| \right)^2 \\
&= |Y| \left(\alpha |X| \right)^2 . \qquad (4.9)
\end{aligned}$$

Dabei ist die Ungleichung genau dann scharf, wenn a und b linear abhängig sind, alle Knoten von Y also denselben Grad haben. □

Das folgende Lemma schätzt die Anzahl Knoten in einer Partition eines bipartiten Graphen ab, deren Nachbarschaftsgröße kleiner als der halbe Durchschnittsgrad ist. Diese Knoten sind keine Kandidaten, um die aktuelle Biclique zu erweitern und kommen in den Mengen $X_i^{(j)}$ daher nicht vor.

Lemma 10. *Sei ein bipartiter Graph $(X \cup Y, E)$ mit $|E| = \alpha|X||Y|$ gegeben. Weiterhin sei $X' = \{x \in X : |N(x)| \geq \frac{1}{2}\alpha|Y|\}$. Dann gilt*

$$|X'| \geq \frac{1}{2}\alpha|X|. \qquad (4.10)$$

Beweis. Jede Kante verläuft entweder zwischen (X', Y) oder $(X \setminus X', Y)$. Damit ergibt sich

$$\begin{aligned}
\alpha|X||Y| = e(X,Y) &= e(X \setminus X', Y) + e(X', Y) \\
&\leq \frac{1}{2}\alpha|X \setminus X'||Y| + e(X', Y) \leq \frac{1}{2}\alpha|X||Y| + e(X', Y), \qquad (4.11)
\end{aligned}$$

also $e(X', Y) \geq \frac{1}{2}\alpha|X||Y|$.

Angenommen, es gilt nun $|X'| < \frac{1}{2}\alpha|X|$. Dann beträgt die Anzahl Kanten zwischen X und Y

$$e(X,Y) < \frac{1}{2}\alpha|X| \cdot |Y| + |X| \cdot \frac{1}{2}\alpha|Y| = \alpha|X||Y| = e(X,Y), \tag{4.12}$$

was ein Widerspruch ist. Es gilt daher $|X'| \geq \frac{1}{2}\alpha|X|$. □

Sei der induzierte Subgraph mit Partitionen X_i und Y_i gegeben. Dann gilt für jeden Knoten aus X_i, dass er weniger als $\alpha_i|Y_i|/2$ Nachbarn hat oder er genau in einer Partition

$$X_i^{(j)} = \{x \in X_i : 2^{j-1}\alpha_i|Y_i| \leq |N_{Y_i}(x)| < 2^j\alpha_i|Y_i|\} \tag{4.13}$$

mit $0 \leq j \leq \lceil \log_2(1/\alpha_i) \rceil$ enthalten ist.

Für den Beweis des folgenden Lemmas wird gerade die Eigenschaft benötigt, dass sich in einer Partition der Minimal- und Maximalgrad höchstens um einen Faktor zwei unterscheiden.

Lemma 11. *Sei ein bipartiter Graph mit Partitionen X_i, Y_i und Dichte α_i sowie $X_i^{(j)}$ gegeben. Dann existiert für jedes $0 \leq j \leq \lceil \log_2(1/\alpha_i) \rceil$ mit $X_i^{(j)} \neq \emptyset$ ein $x_i \in X_i^{(j)}$ mit*

$$e(X_i^{(j)}, N_{Y_i}(x_i)) \geq \frac{\alpha_i}{4} \cdot |X_i^{(j)}| \cdot |N_{Y_i}(x_i)|. \tag{4.14}$$

Beweis. Es gilt

$$\begin{aligned}
\sum_{x \in X_i^{(j)}} e(X_i^{(j)}, N_{Y_i}(x)) &= \sum_{x \in X_i^{(j)}} \sum_{y \in N_{Y_i}(x)} |N(y) \cap X_i^{(j)}| \\
&= \sum_{y \in Y_i} |N(y) \cap X_i^{(j)}|^2 \\
&\underset{\text{L.9}}{\geq} \frac{(e(X_i^{(j)}, Y_i))^2}{|Y_i|} \\
&\geq \frac{(2^{j-1}\alpha_i|Y_i||X_i^{(j)}|)^2}{|Y_i|} \\
&= 2^{2j-2}\alpha_i^2|Y_i||X_i^{(j)}|^2.
\end{aligned} \tag{4.15}$$

Durch die Bildung des arithmetischen Mittelwertes und der Ungleichung $|N_{Y_i}(x)| \leq 2^j\alpha_i|Y_i|$ folgt die Existenz eines $x \in X_i^{(j)}$ mit

$$\begin{aligned}
x \in X_i^{(j)} : \quad e(X_i^{(j)}, N_{Y_i}(x)) &\geq 2^{2j-2}\alpha_i^2 \cdot |Y_i| \cdot |X_i^{(j)}| \\
&\geq 2^{j-2}\alpha_i \cdot |N_{Y_i}(x)| \cdot |X_i^{(j)}| \\
&\geq \frac{\alpha_i}{4} \cdot |N_{Y_i}(x)| \cdot |X_i^{(j)}|.
\end{aligned} \tag{4.16}$$

□

Diese Abschätzung gilt also auch für das j, das $X_i^{(j)}$ maximiert und vom Algorithmus gewählt wird. Bei der nächsten Iteration $(X_{i+1}, Y_{i+1}) := (X_i^{(j)} \setminus \{x\}, N_{Y_i}(x))$ gilt dann für die Dichte α_{i+1}

$$\begin{aligned}
\alpha_{i+1} &= \frac{e(X_i^{(j)}, N_{Y_i}(x)) - |N_{Y_i}(x)|}{(|X_i^{(j)}| - 1) \cdot |N_{Y_i}(x)|} \\
&\geq \frac{\frac{\alpha_i}{4} \cdot |X_i^{(j)}| \cdot |N_{Y_i}(x)| - |N_{Y_i}(x)|}{(|X_i^{(j)}| - 1) \cdot |N_{Y_i}(x)|} \\
&= \frac{\frac{\alpha_i}{4} \cdot |X_i^{(j)}| - 1}{|X_i^{(j)}| - 1} \\
&\geq \frac{\alpha_i}{8}, \qquad (4.17)
\end{aligned}$$

sofern $|X_i^{(j)}|$ bei der letzten Abschätzung hinreichend groß ist. Aus dieser Rekursionsungleichung folgt dann für die Dichte nach dem i-ten Schritt die explizite Darstellung

$$\alpha_i \geq \frac{1}{8^i} \cdot \alpha_0. \qquad (4.18)$$

Für die Größe von Y_{i+1} gilt die Abschätzung

$$\begin{aligned}
|Y_{i+1}| &= |N_{Y_i}(x_{i+1})| \\
&\underset{\text{L.10}}{\geq} \frac{\alpha_i}{2} \cdot |Y_i| \\
&\geq |Y_0| \cdot \prod_{j=0}^{i} \frac{\alpha_j}{2} \\
&\geq |Y_0| \cdot \prod_{j=0}^{i} \frac{(1/8)^j \alpha_0}{2} \\
&= |Y_0| \cdot \left(\frac{\alpha_0}{2}\right)^{i+1} \cdot \left(\frac{1}{8}\right)^{\sum_{j=0}^{i} j} \\
&= |Y_0| \cdot \left(\frac{\alpha_0}{2}\right)^{i+1} \cdot \left(\frac{1}{8}\right)^{i(i+1)/2} \\
&=: Y(i, \alpha_0). \qquad (4.19)
\end{aligned}$$

Nach Lemma 10 gibt es in X_i mindestens $|X_i|\alpha_i/2$ Knoten mit einem Grad größer als $|Y_i|\alpha_i/2$. Außerdem gibt es $\lceil \log_2(1/\alpha_i) \rceil + 1$ Partitionen $X_i^{(j)}$. Da

der Algorithmus für X_{i+1} eine kardinalitätsmaximale Partition $X_i^{(j)}$ wählt, ergibt sich für $|X_i|$ die Abschätzung

$$\begin{aligned}
|X_{i+1}| &\geq \frac{\alpha_i}{2} \cdot \frac{|X_i|}{\lceil \log_2(1/\alpha_i) \rceil + 1} - 1 \\
&\geq \frac{\alpha_i}{3\lceil \log_2(1/\alpha_i) \rceil} \cdot |X_i| \\
&\geq \prod_{j=0}^{i} \frac{\alpha_j}{3\lceil \log_2(1/\alpha_j) \rceil} \cdot |X_0| \\
&= \frac{\prod_{j=0}^{i} \alpha_j}{\prod_{j=0}^{i} 3\lceil \log_2(1/\alpha_j) \rceil} \cdot |X_0| \\
&\geq \frac{\prod_{j=0}^{i} (1/8)^j \alpha_0}{3^{i+1} \prod_{j=0}^{i} \lceil \log_2(1/\alpha_j) \rceil} \cdot |X_0| \\
&= \frac{\alpha_0^{i+1} (1/8)^{\sum_{j=0}^{i} j}}{3^{i+1} \prod_{j=0}^{i} \lceil \log_2(1/\alpha_j) \rceil} \cdot |X_0| \\
&= \left(\frac{\alpha_0}{3}\right)^{i+1} \cdot \frac{(1/8)^{i(i+1)/2}}{\prod_{j=0}^{i} \lceil \log_2(1/\alpha_j) \rceil} \cdot |X_0|. \qquad (4.20)
\end{aligned}$$

Der Term $\lceil \log_2(1/\alpha_j) \rceil$ kann wegen (4.18) durch $\lceil \log_2(1/(\frac{\alpha_0}{8^i})) \rceil$ nach oben abgeschätzt werden, so dass

$$|X_{i+1}| \geq \left(\frac{\alpha_0}{3}\right)^{i+1} \cdot \frac{(1/8)^{i(i+1)/2}}{(\lceil \log_2(8^i/\alpha_0) \rceil)^{i+1}} \cdot |X_0| =: X(i, \alpha_0) \qquad (4.21)$$

gilt.

Satz 15. *Zu jedem $\alpha \in (0,1)$ existiert ein n_0, so dass für jeden bipartiten Graphen $(X \cup Y, E)$ mit $|X| = |Y| \geq n_0$ und $\frac{|E|}{|X||Y|} \geq \alpha$ der Algorithmus eine balancierte Biclique der Größe $(1 - o(1)) \cdot \sqrt{2\log(|X_0|)}$ berechnet.*

Beweis. Im Initialisierungsschritt gilt $X_0 := X$, $Y_0 := Y$ und $\alpha_0 := \alpha$.

Im Folgenden wird die Anzahl Iterationen, die mindestens möglich sind, nach unten abgeschätzt. Dazu sei i_y die kleinste natürliche Zahl mit $i_y > |Y_{i_y}|$ und i_x die kleinste natürliche Zahl mit $|X_{i_x}| < 1$. Dann ist $i' := \min\{i_x, i_y\} - 1$ die Anzahl Iterationen, die der Algorithmus ausführt und folglich die Größe der vom Algorithmus zurückgegebenen Biclique. Um i' nach unten abzuschätzen wird zunächst i_y nach unten abgeschätzt.

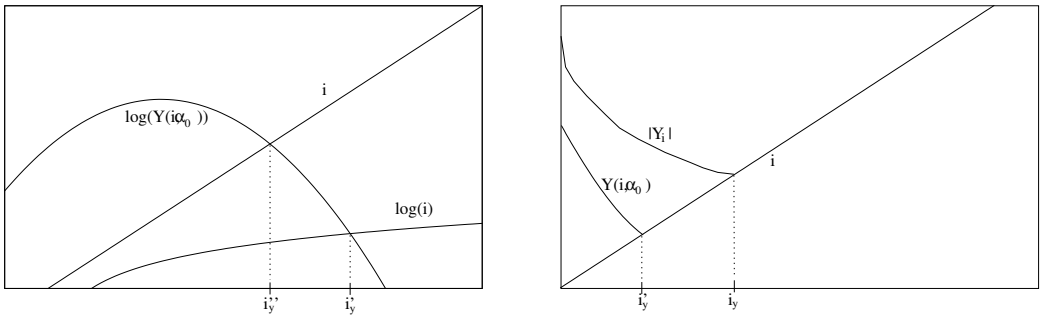


Abbildung 4.3: *Skizzen zur unteren Abschätzung von i_y*

Wegen (4.19) ist $i'_y := \min\{i : i > Y(i, \alpha_0)\}$ eine untere Schranke für i_y (s. Abbildung 4.3, rechts). Logarithmieren der Ungleichung $Y(i, \alpha_0) < i$ und der Abschätzung[2] $log(i) < i$ führt auf

$$i \cdot \log(\alpha_0/2) + \frac{i(i-1)}{2} \cdot \log(1/8) + \log(|Y_0|) < i. \tag{4.22}$$

Dann ist $i''_y := \min_{i>0}\{i : \log(Y(i, \alpha_0)) < i\}$ eine untere Schranke für i'_y (s. Abbildung 4.3, links). Die linke Seite von (4.22) ist eine nach unten geöffnete Parabel mit der Lösung[3]

$$i''_y = \frac{f(\alpha_0) + \sqrt{(f(\alpha_0))^2 + 8\log(8)\log(|Y_0|)}}{2\log(8)}, \tag{4.23}$$

wobei $f(\alpha_0) := 2\log\left(\frac{\alpha_0}{2}\right) + \log(8) - 2$ ist. Zu jedem α_0 und für hinreichend großes $|Y_0|$ gilt dann

$$i_y > i''_y = (1 - o(1)) \cdot \sqrt{2\log(|Y_0|)}. \tag{4.24}$$

Auf ähnliche Weise kann i_x nach unten abgeschätzt werden. $i'_x := \min_{i>0}\{i : X(i, \alpha_0) < 1\}$ ist wegen (4.21) zunächst eine untere Schranke für i_x (s. Abbildung 4.4, rechts).

Logarithmieren von $X(i, \alpha_0) < 1$ auf beiden Seiten führt auf

$$i\log\left(\frac{\alpha_0}{3}\right) + \frac{i(i-1)}{2}\log\left(\frac{1}{8}\right) - i\log\left(\lceil\log_2\left(\frac{8^i}{\alpha_0}\right)\rceil\right) + \log(|X_0|) < 0. \tag{4.25}$$

Sofern keine Biclique der Größe mindestens $\log(|X_0|)$ berechnet wird, gilt die Abschätzung $i < \log(|X_0|)$. Unter Verwendung dieser Abschätzung und Umformung von (4.25) ergibt sich der Term

[2] Abschätzung gilt für jede Basis größer als $e^{1/e} \doteq 1.4446$

[3] Die andere Lösung ist negativ und hier nicht relevant.

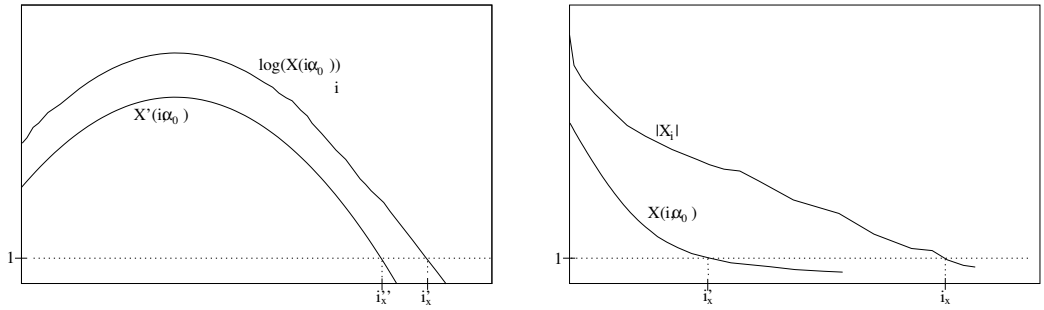


Abbildung 4.4: *Skizzen zur unteren Abschätzung von i_x*

$$X'(i,\alpha_0) := \log(|X_0|) + \frac{i^2}{2}\log\left(\frac{1}{8}\right) + \\ + i\left(\log\left(\frac{8^2\alpha_0}{3}\right) - \log\left(\lceil\log_2\left(\frac{8^{\log(|X_0|)}}{\alpha_0}\right)\rceil\right)\right), \qquad (4.26)$$

wobei $X'(i,\alpha_0) < \log(X(i,\alpha_0))$ für $1 < i < \log(|X_0|)$ gilt.

Dann ist $i''_x := \min_{i>0}\{i : X'(i,\alpha_0) < 0\}$ eine untere Schranke für i'_x (s. Abbildung 4.4, links). $X'(i,\alpha_0)$ ist wiederum eine nach unten geöffnete Parabel, und für i''_x ergibt sich daher die Lösung

$$i''_x = \frac{g(\alpha_0,|X_0|) + \sqrt{(g(\alpha_0,|X_0|))^2 + 8\log(8)\log(|X_0|)}}{2\log(8)}, \qquad (4.27)$$

wobei $g(\alpha_0,|X_0|) := 2\log\left(\frac{\alpha_0}{3}\right) + \log(8) - 2 - 2\log\left(\lceil\log_2\left(\frac{8^{\log(|X_0|)}}{\alpha_0}\right)\rceil\right)$ ist.

Für konstantes α_0 gilt $(g(\alpha_0,|X_0|))^2 = o(\sqrt{\log(|X_0|)})$. Daher ist im Zähler $\sqrt{8\log(8)\log(|X_0|)}$ der dominierende Term und zu jedem α_0 und für hinreichend großes $|X_0|$ gilt daher

$$i_x > i''_x = (1-o(1))\cdot\sqrt{2\log(|X_0|)}. \qquad (4.28)$$

In Verbindung mit (4.24) folgt daher

$$i' = \min\{i_x,i_y\} - 1 \geq \min\{i''_x,i''_y\} = (1-o(1))\cdot\sqrt{2\log(|X_0|)}. \qquad (4.29)$$

□

4.3.3 Laufzeit des Algorithmus

In diesem Abschnitt wird auf die Laufzeit des im vorangegangenen Abschnittes analysierten Algorithmus eingegangen.

Satz 16. *Sei $G = (X \cup Y, E)$ ein bipartiter Graph mit $|X| = |Y|$. Dann hat der balancierte Bicliquenalgorithmus aus Abbildung 4.2 eine Laufzeit von $\mathcal{O}(|X||E|)$.*

Beweis. Ohne Einschränkung sei $|E| \geq |X|$ und der Graph als Adjazenzliste gespeichert. Die folgenden Zeilenangaben beziehen sich auf Abbildung 4.2. Die While-Schleife in Zeile 2 wird maximal $|X|$ durchlaufen. Es bleibt zu zeigen, dass jeder Schritt innerhalb der Schleife eine Laufzeit von $\mathcal{O}(|E| + |X|) = \mathcal{O}(|E|)$ hat. Seien die Partitionen X_i und Y_i gegeben. Das Bestimmen einer kardinaltätsmaximalen Menge $X_i^{(j)}$ in Zeile 3 und 4 ist offensichtlich in $\mathcal{O}(|E| + |X|) = \mathcal{O}(|E|)$ möglich, ebenso die Konstruktion des induzierten Graphen mit Partitionen $X_i^{(j)}$ und Y_i.

Unter Verwendung der Gleichung

$$e((X_i^{(j)}), N_{Y_i}(x)) = \sum_{y \in N_{Y_i}(x)} |N(y) \cap X_i^{(j)}| \tag{4.30}$$

kann für alle $x \in X_i^{(j)}$ der Wert $e((X_i^{(j)}), N_{Y_i}(x))$ in linearer Zeit durch die folgenden Schleifen und einem Hilfsfeld A bestimmt werden.

`for each` $x \in X_i^{(j)}$: $A[x] = 0$
`for each` $y \in Y_i$
    `for each` $x \in N(y) \cap X_i^{(j)}$
        $A[x] = A[x] + | N(y) \cap X_i^{(j)} |$

Für alle $x \in X_i^{(j)}$ gilt dann $A[x] = e((X_i^{(j)}), N_{Y_i}(x))$ bei einer Laufzeit von $\sum_{y \in Y}(d(y) + \mathcal{O}(1)) = \mathcal{O}(|E| + |Y|) = \mathcal{O}(|E|)$.

Das Auswählen eines geeigneten x_{i+1} in Zeile 4, sowie das Bestimmen des neuen Subgraphen mit den Partitionen X_{i+1} und Y_{i+1} ist ebenfalls in $\mathcal{O}(|E|)$ möglich. Damit ergibt sich eine Gesamtlaufzeit von $\mathcal{O}(|X||E|)$. □

4.3.4 Approximation auf $\log(n)$ möglich?

Es bleibt die Frage offen, ob in bipartiten Graphen mit Dichte α eine Biclique der Größe $k \log_{1/\alpha}(n)$ mit geeignetem konstanten k in polynomieller Zeit gefunden werden kann. Der Hauptgrund, warum bei dem hier vorgestellten Algorithmus die Analyse nur $\Theta(\sqrt{\log(n)})$ Iterationen garantiert, besteht in der Abnahme der Dichte um einen konstanten Faktor pro Iteration, vgl. dazu

die Gleichung (4.17). Als Folge tritt bei der Abschätzung für $|X_i|$ und $|Y_i|$ der quadratische Exponent $(1/8)^{\binom{i+1}{2}}$ in (4.19) und (4.20) auf. Sofern die Dichte pro Iteration nicht abnimmt, entfällt dieser Faktor und eine logarithmische Anzahl an Iterationen wäre aufgrund der Faktoren $(\alpha/2)^{i+1}$ bzw. $(\alpha/3)^{i+1}$ möglich.

Eng verwandt und auch eine strukturell interessante Frage ist die Folgende.

> Sei ein bipartiter Graph mit $(X \cup Y, E)$ mit Dichte α gegeben. Existiert stets ein $x \in X$ mit den zwei Eigenschaften:
>
> (i) Die Dichte im induzierten Graphen mit den Partitionen $N(x)$ und $\bigcup_{y \in N(x)} N(y)$ beträgt mindestens α;
>
> (ii) Es gilt $|N(x)| \geq \frac{\alpha}{2}|Y|$ und $|\bigcup_{y \in N(x)} N(y)| \geq \frac{\alpha}{2}|X|$?

Die erste Eigenschaft fordert, dass die Dichte nicht abnimmt, wohingegen die zweite Eigenschaft garantiert, dass die beiden neuen Partitionen im nächsten Iterationsschritt nicht zu klein werden.[4] Bei zufälligen bipartiten Graphen mit Dichte α ist diese Forderung mit hoher Wahrscheinlichkeit stets erfüllt, und daher lässt sich mit hoher Wahrscheinlichkeit auch eine Biclique der Größe $(1 - o(1)) \log_{1/\alpha}(n)$ mit diesem Greedyansatz konstruieren.

[4] Anstelle des Faktors 1/2 kann auch ein beliebig anderer konstanter Faktor genommen werden.

Teil III

Anhang A

Ungleichungen zu den Instanzen vom Loss-Algorithmus

In diesem Anhang befindet sich das Gleichungssystem zu dem Beweis aus Satz 9. Zuerst folgt die Zielfunktion, wobei der minimale Steinerbaum mit eins normiert ist. Die nächsten fünf Ungleichungen stammen aus (2.38) und (2.39). Die restlichen Ungleichungen forcieren, dass zum einen der minimal spannende Baum im Distanzgraphen im Ausgangsgraph durch die horizontalen Kanten induziert wird und zum anderen stets a_i als Loss gewählt wird und nicht b_i. Die gerundete Güte der Instanz beträgt 1.23389.

```
NMaximize[{(a1 + a2 + a3 + a4 + 4*(b1 + b2 + b3 + b4))/20,
    a1*(4) <= 8 - 4*b1, a2*(2 + b1) <= 8 - 4*b2,
    a3*(0 + b1 + b2) <= 8 - 4*b3,
    a4*(-2 + b1 + b2 + b3) <= 8 - 4*b4, b1 + b2 + b3 + b4 <= 5,
    a1 + b1 >= 2, a2 + b2 >= 2, a3 + b3 >= 2, a4 + b4 >= 2,
    0 <= a1 <= b1 <= 2, 0 <= a2 <= b2 <= 2,
    0 <= a3 <= b3 <= 2, 0 <= a4 <= b4 <= 2},
   {a1, b1, a2, b2, a3, b3, a4, b4}]

{1.23389, {a1 -> 1., a2 -> 1.14286, a3 -> 1.2675, a4 -> 1.26746,
    b1 -> 1., b2 -> 1.14286, b3 -> 1.32098, b4 -> 1.53616}}
```

Bei diesem Ungleichungssystem ist zusätzlich die Bedingung $a_i = b_i$ eingearbeitet und dann entsprechend vereinfacht worden. Die Lösung liefert daher für die quasibipartite Instanz aus Korollar 5 eine Kantenbelegung mit einer Güte von $\frac{2424451}{1973356}$.

```
Maximize[{(a1 + a2 + a3 + a4 + 4*(a1 + a2 + a3 + a4))/20,
  a1*(4) <= 8 - 4*a1, a2*(2 + a1) <= 8 - 4*a2,
  a3*(0 + a1 + a2) <= 8 - 4*a3, a4*(-2 + a1 + a2 + a3) <= 8 - 4*a4,
  a1 + a2 + a3 + a4 <= 5,
  a1 + a1 >= 2, a2 + a2 >= 2, a3 + a3 >= 2, a4 + a4 >= 2,
  0 <= a1 <= 2, 0 <= a2 <= 2, 0 <= a3 <= 2, 0 <= a4 <= 2},
 {a1, a2, a3, a4}]

{2424451/1973356,
  {a1 -> 1, a2 -> 8/7, a3 -> 56/43, a4 -> 2408/1639}}
```

Anhang B

Ungleichungen zu den Instanzen des MSS-Algorithmus

In diesem Anhang befindet sich das Gleichungssystem zu dem Beweis aus Satz 10. Zuerst folgt die Zielfunktion, dann die Länge des normierten minimalen Steinerbaums. Die anschließenden sechs Ungleichungen stammen aus (2.41).

```
Maximize[{5 (x1 + x2 + x3) + 4*(a + b), 4*4*a + 4*b == 1,
  5/4*x1 <= (4*a + b)/4, 5/4*x1 <= 4*a/3, 5/4*x2 <= (3*a + b)/3,
  5/4*x2 <= 3*a/2, 5/4*x3 <= (2*a + b)/2, 5/4*x3 <= 2*a/1,
  0 <= x1 <= x2 <= x3, 0 <= a, 0 <= b}, {a, b, x1, x2, x3}]
```

```
{47/36, {a -> 1/24, b -> 1/12, x1 -> 2/45, x2 -> 1/20, x3 -> 1/15}}
```

Hier sind die Ungleichungen von Abbildung 2.15 auf Seite 46 abgebildet. Nach der Zielfunktion und der Normierung für die Länge des minimalen Steinerbaums, sorgen die Ungleichungen dafür, dass hintereinander die Komponenten $\{1,2\}$, $\{3,4\}$, $\{5,6\}$, $\{1,3\}$ und $\{3,5\}$ gewählt werden können, indem im jeweiligen Schritt die entsprechende Komponente die Auswahlfunktion minimiert.

```
Maximize[{a + b + c + d + e + f + a + g + h + c + c + h + i + e,
    a + b + c + d + e + f + g + h + i == 1, a + b <= a + g + h + c,
    a + b <= a + g + i + e, a + b <= c + d, a + b <= c + h + i + e,
    a + b <= e + f, a + b <= (a + b + c + g + h)/2,
    a + b <= (a + b + e + g + i)/2, a + b <= (a + c + d + g + h)/2,
    a + b <= (a + c + e + g + h + i)/2,
```

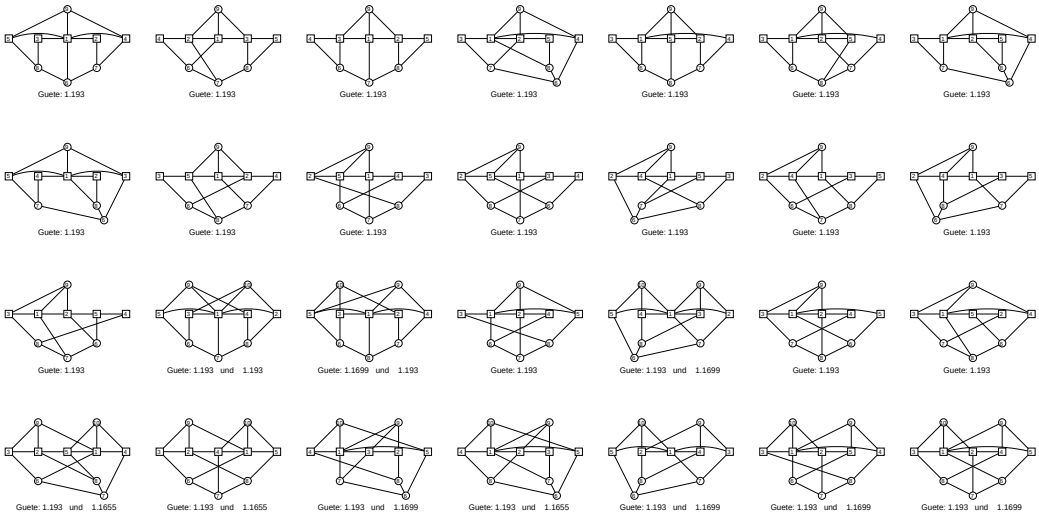
```
    a + b <= (a + e + f + g + i)/2,
    a + b <= (c + d + e + h + i)/2, a + b <= (c + e + f + h + i)/2,
    a + b <= (a + b + c + d + g + h)/3,
    a + b <= (a + b + c + e + g + h + i)/3,
    a + b <= (a + b + e + f + g + i)/3,
    a + b <= (a + c + d + e + g + h + i)/3,
    a + b <= (a + c + e + f + g + h + i)/3,
    a + b <= (c + d + e + f + h + i)/3,
    a + b <= (a + b + c + d + e + g + h + i)/4,
    a + b <= (a + b + c + e + f + g + h + i)/4,
    a + b <= (a + c + d + e + f + g + h + i)/4,
    a + b <= (a + b + c + d + e + f + g + h + i)/5,
    c + d <= a + g + h + c,
    c + d <= a + g + i + e, c + d <= c + h + i + e, c + d <= e + f,
    c + d <= (a + c + d + g + h)/2,
    c + d <= (a + c + e + g + h + i)/2,
    c + d <= (a + e + f + g + i)/2, c + d <= (c + d + e + h + i)/2,
    c + d <= (c + e + f + h + i)/2,
    c + d <= (a + c + d + e + g + h + i)/3,
    c + d <= (a + c + e + f + g + h + i)/3,
    c + d <= (c + d + e + f + h + i)/3,
    c + d <= (a + c + d + e + f + g + h + i)/4,
    e + f <= a + g + h + c,
    e + f <= a + g + i + e,
    e + f <= c + h + i + e,
    e + f <= (a + c + e + g + h + i)/2,
    e + f <= (a + e + f + g + i)/2,
    e + f <= (c + e + f + h + i)/2,
    e + f <= (a + c + e + f + g + h + i)/3,
    a + c + g + h <= a + g + i + e,
    a + c + g + h <= c + h + i + e,
    a + c + g + h <= (a + c + e + g + h + i)/2,
    c + e + h + i <= a + g + i + e,
    0 <= a <= b, 0 <= c <= d, 0 <= e <= f, 0 <= g, 0 <= h, 0 <= i},
    {a, b, c, d, e, f, g, h, i}]

3/2, {a -> 4/45, b -> 4/45, c -> 1/9, d -> 1/9, e -> 2/15,
 f -> 2/15, g -> 7/90, h -> 1/18, i -> 1/5}
```

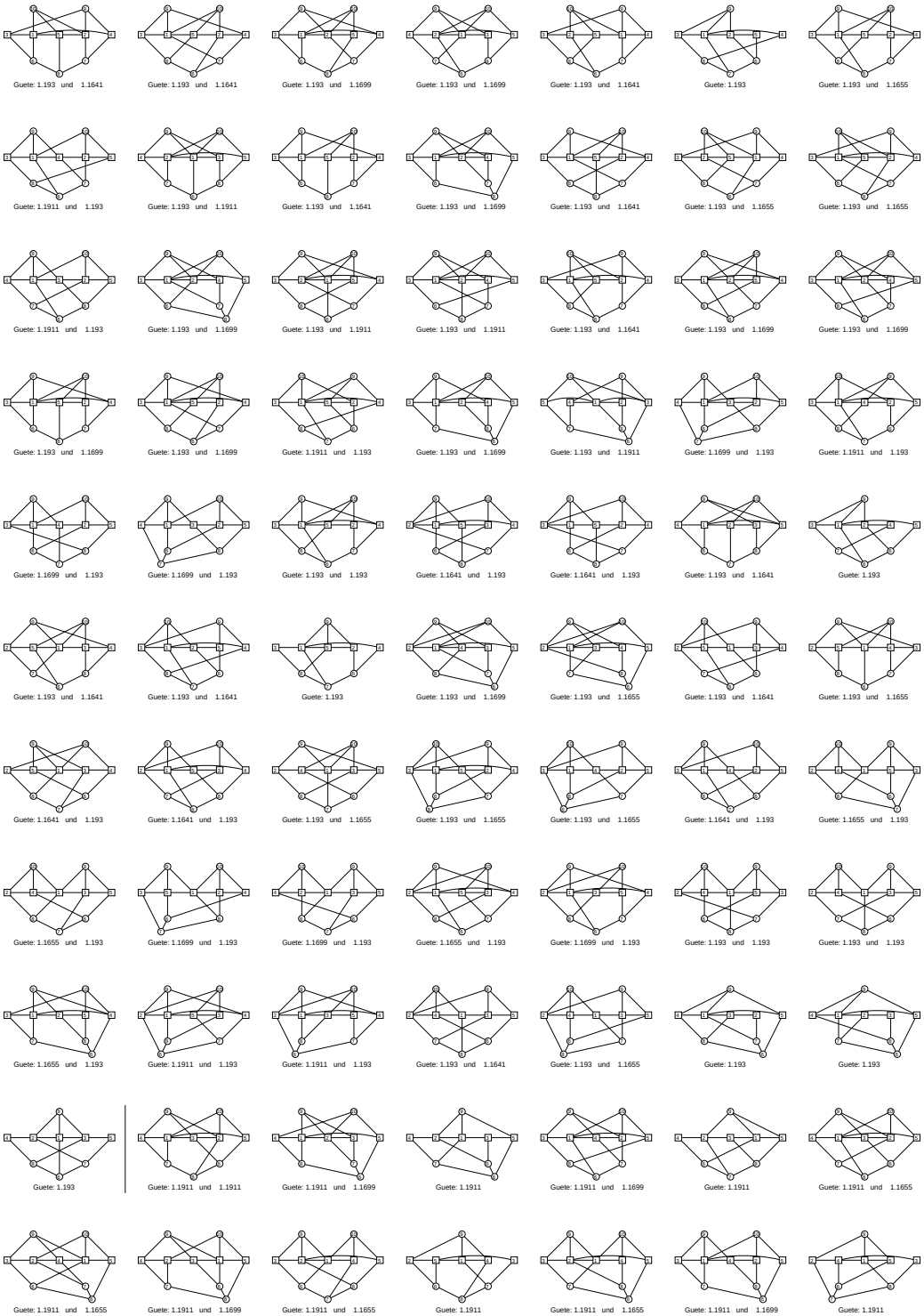
Anhang C

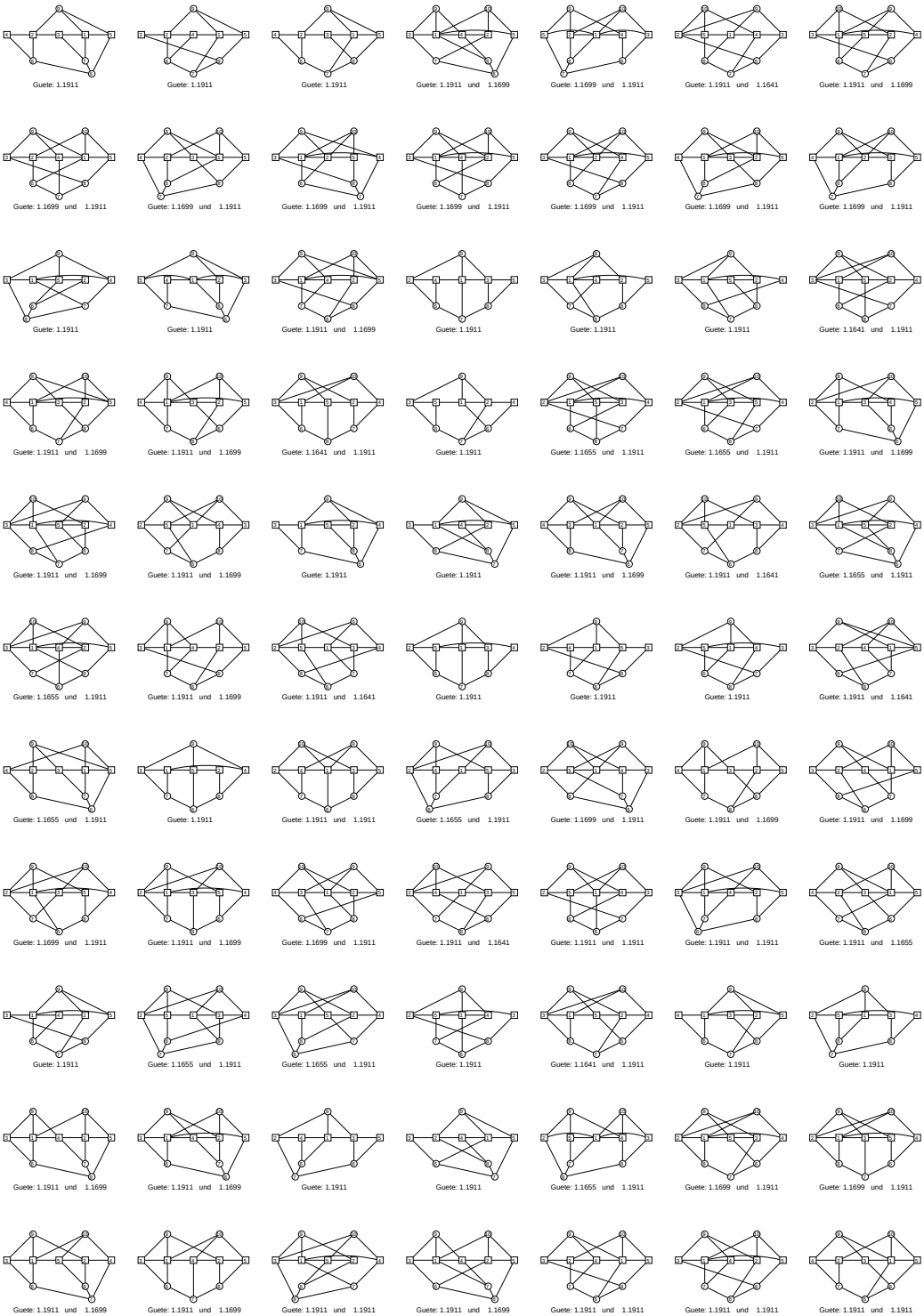
Die Topologien $\mathcal{G}_5$ für den Relativen Greedy Algorithmus

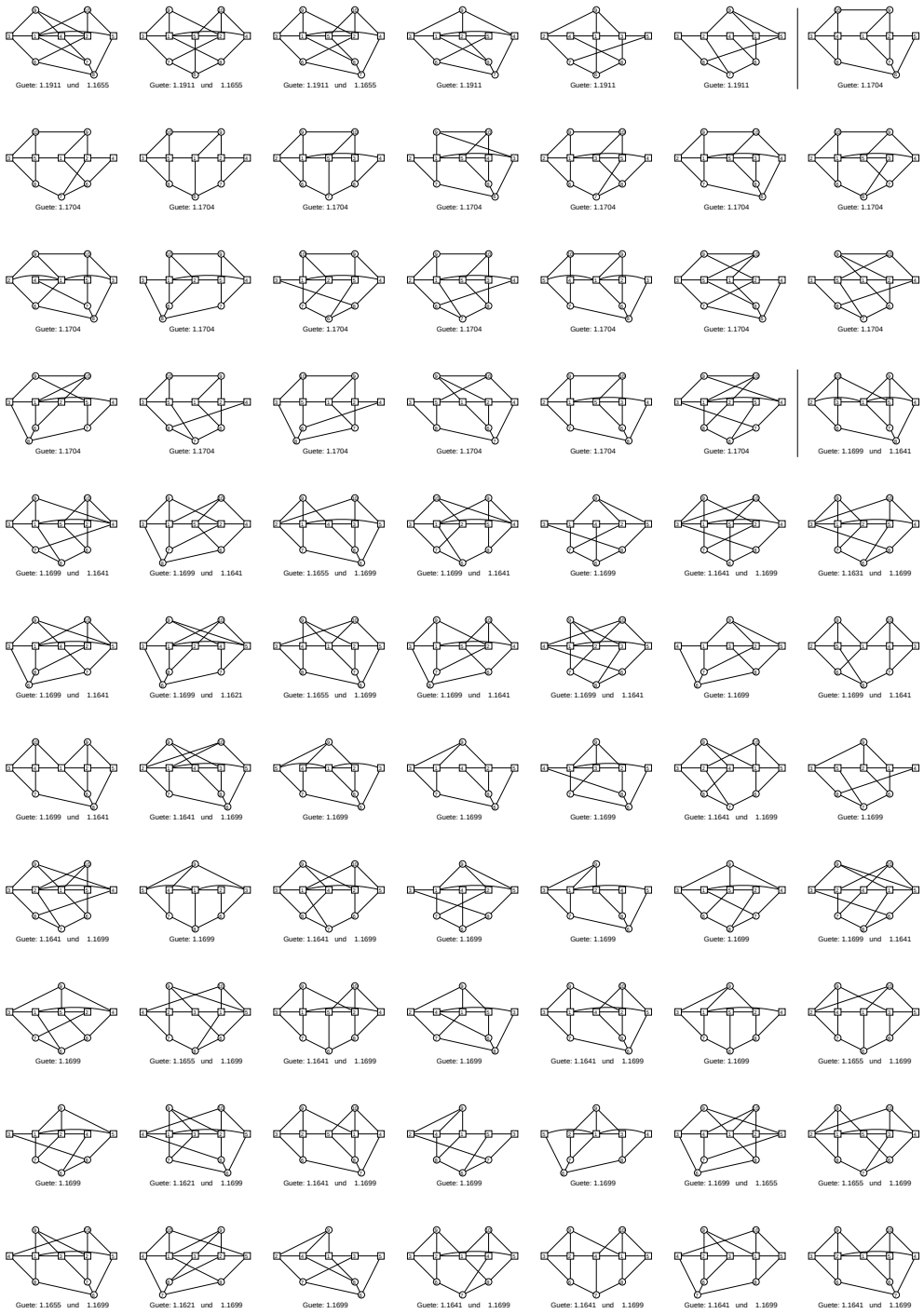
Abgebildet sind alle 663 Topologien für den Relativen Greedy Algorithmus mit fünf Terminalen und in ihrer Güte absteigend sortiert. Die erste bzw. (sofern vorhandene) zweite Zahl x entspricht bei geeigneter Kantenbelegung eine erreichbare Güte, wenn die Nachbarschaft von Knoten 9 bzw. 10 im ersten Schritt ausgewählt wird. Bei einem vertikalen Strich zwischen zwei Instanzen fängt eine neue Güte an (vgl. dazu Tabelle 2.4).

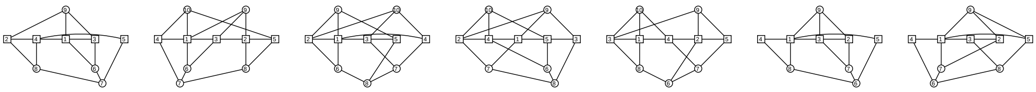
Die Genauigkeit liegt bei $\epsilon = 0.01$, d.h. es gibt keine Kantenbelegung mit einer Güte von mehr als $(1 + \epsilon)x$.



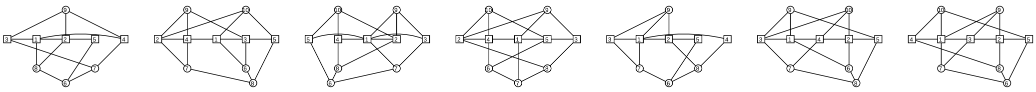

Guete: 1.193 und 1.1641
Guete: 1.193 und 1.1641
Guete: 1.193 und 1.1699
Guete: 1.193 und 1.1699
Guete: 1.193 und 1.1641
Guete: 1.193
Guete: 1.193 und 1.1655
Guete: 1.1911 und 1.193
Guete: 1.193 und 1.1911
Guete: 1.193 und 1.1641
Guete: 1.193 und 1.1699
Guete: 1.193 und 1.1641
Guete: 1.193 und 1.1655
Guete: 1.193 und 1.1655
Guete: 1.1911 und 1.193
Guete: 1.193 und 1.1699
Guete: 1.193 und 1.1911
Guete: 1.193 und 1.1911
Guete: 1.193 und 1.1641
Guete: 1.193 und 1.1699
Guete: 1.193 und 1.1699
Guete: 1.193 und 1.1699
Guete: 1.193 und 1.1699
Guete: 1.1911 und 1.193
Guete: 1.193 und 1.1699
Guete: 1.193 und 1.1911
Guete: 1.1699 und 1.193
Guete: 1.1911 und 1.193
Guete: 1.1699 und 1.193
Guete: 1.1699 und 1.193
Guete: 1.193 und 1.193
Guete: 1.1641 und 1.193
Guete: 1.1641 und 1.193
Guete: 1.193 und 1.1641
Guete: 1.193
Guete: 1.193 und 1.1641
Guete: 1.193 und 1.1641
Guete: 1.193
Guete: 1.193 und 1.1699
Guete: 1.193 und 1.1655
Guete: 1.193 und 1.1641
Guete: 1.193 und 1.1655
Guete: 1.1641 und 1.193
Guete: 1.1641 und 1.193
Guete: 1.193 und 1.1655
Guete: 1.193 und 1.1655
Guete: 1.193 und 1.1655
Guete: 1.1641 und 1.193
Guete: 1.1655 und 1.193
Guete: 1.1655 und 1.193
Guete: 1.1699 und 1.193
Guete: 1.1699 und 1.193
Guete: 1.1655 und 1.193
Guete: 1.1699 und 1.193
Guete: 1.193 und 1.193
Guete: 1.193 und 1.193
Guete: 1.1655 und 1.193
Guete: 1.1911 und 1.193
Guete: 1.1911 und 1.193
Guete: 1.193 und 1.1641
Guete: 1.193 und 1.1655
Guete: 1.193
Guete: 1.193
Guete: 1.193
Guete: 1.1911 und 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911
Guete: 1.1911 und 1.1655
Guete: 1.1911 und 1.1655
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1655
Guete: 1.1911
Guete: 1.1911 und 1.1655
Guete: 1.1911 und 1.1699
Guete: 1.1911


Guete: 1.1911
Guete: 1.1911
Guete: 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1699 und 1.1911
Guete: 1.1911 und 1.1641
Guete: 1.1911 und 1.1699
Guete: 1.1699 und 1.1911
Guete: 1.1699 und 1.1911
Guete: 1.1699 und 1.1911
Guete: 1.1699 und 1.1911
Guete: 1.1699 und 1.1911
Guete: 1.1699 und 1.1911
Guete: 1.1911
Guete: 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911
Guete: 1.1911
Guete: 1.1911
Guete: 1.1641 und 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1699
Guete: 1.1641 und 1.1911
Guete: 1.1911
Guete: 1.1655 und 1.1911
Guete: 1.1655 und 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1699
Guete: 1.1911
Guete: 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1641
Guete: 1.1655 und 1.1911
Guete: 1.1655 und 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1641
Guete: 1.1911
Guete: 1.1911
Guete: 1.1911 und 1.1641
Guete: 1.1655 und 1.1911
Guete: 1.1911
Guete: 1.1911 und 1.1911
Guete: 1.1655 und 1.1911
Guete: 1.1699 und 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1699
Guete: 1.1699 und 1.1911
Guete: 1.1699 und 1.1911
Guete: 1.1911 und 1.1641
Guete: 1.1911 und 1.1911
Guete: 1.1911 und 1.1655
Guete: 1.1911
Guete: 1.1655 und 1.1911
Guete: 1.1655 und 1.1911
Guete: 1.1911
Guete: 1.1641 und 1.1911
Guete: 1.1911
Guete: 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1699
Guete: 1.1911
Guete: 1.1911
Guete: 1.1655 und 1.1911
Guete: 1.1699 und 1.1911
Guete: 1.1699 und 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1911
Guete: 1.1911 und 1.1699
Guete: 1.1911 und 1.1911
Guete: 1.1911 und 1.1911
Guete: 1.1911 und 1.1911

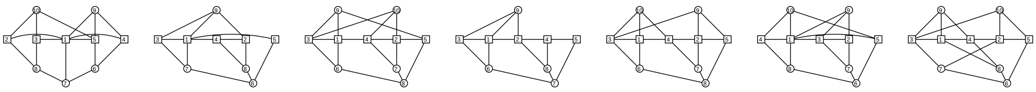

Guete: 1.1911 und 1.1655
Guete: 1.1911 und 1.1655
Guete: 1.1911 und 1.1655
Guete: 1.1911
Guete: 1.1911
Guete: 1.1911
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1704
Guete: 1.1699 und 1.1641
Guete: 1.1699 und 1.1641
Guete: 1.1699 und 1.1641
Guete: 1.1655 und 1.1699
Guete: 1.1699 und 1.1641
Guete: 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1631 und 1.1699
Guete: 1.1699 und 1.1641
Guete: 1.1699 und 1.1621
Guete: 1.1655 und 1.1699
Guete: 1.1699 und 1.1641
Guete: 1.1699 und 1.1641
Guete: 1.1699
Guete: 1.1699 und 1.1641
Guete: 1.1699 und 1.1641
Guete: 1.1641 und 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699 und 1.1641
Guete: 1.1699
Guete: 1.1655 und 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1699
Guete: 1.1655 und 1.1699
Guete: 1.1699
Guete: 1.1621 und 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699 und 1.1655
Guete: 1.1655 und 1.1699
Guete: 1.1655 und 1.1699
Guete: 1.1621 und 1.1699
Guete: 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1641 und 1.1699



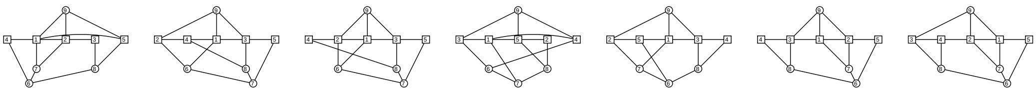
Guete: 1.1699
Guete: 1.1655 und 1.1699
Guete: 1.1699 und 1.1641
Guete: 1.1699 und 1.1631
Guete: 1.1699 und 1.1655
Guete: 1.1699
Guete: 1.1699



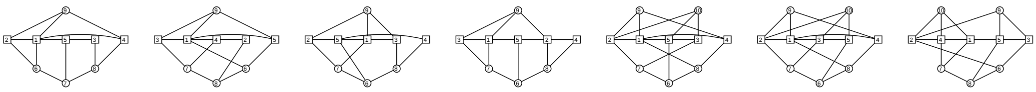
Guete: 1.1699
Guete: 1.1655 und 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1699 und 1.1621
Guete: 1.1699
Guete: 1.1699 und 1.1655
Guete: 1.1655 und 1.1699



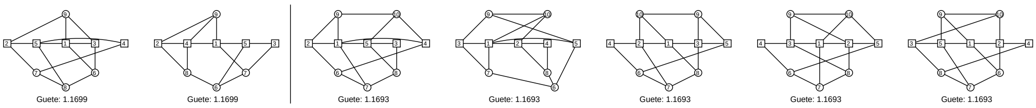
Guete: 1.1699 und 1.1641
Guete: 1.1699
Guete: 1.1699 und 1.1631
Guete: 1.1699
Guete: 1.1699 und 1.1655
Guete: 1.1641 und 1.1699
Guete: 1.1641 und 1.1699



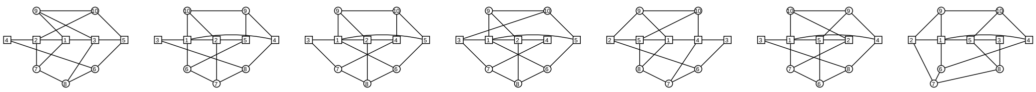
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699



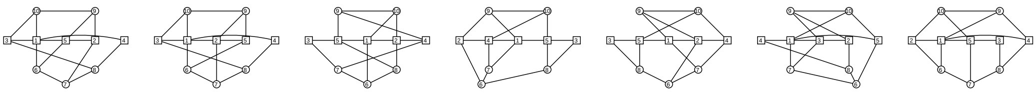
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1641 und 1.1699
Guete: 1.1699 und 1.1641



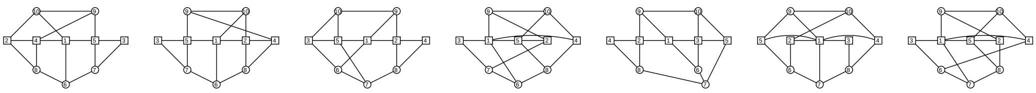
Guete: 1.1699
Guete: 1.1699
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693



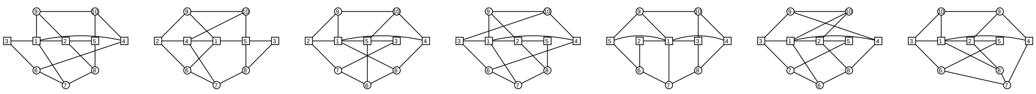
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693



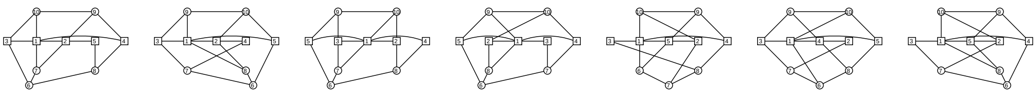
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693



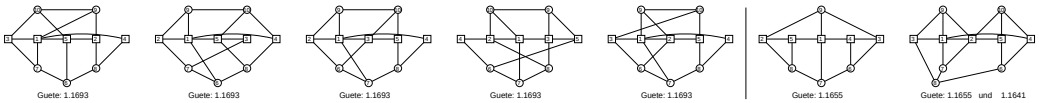
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693

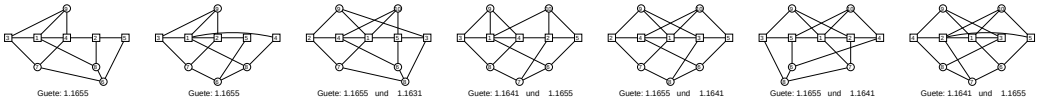


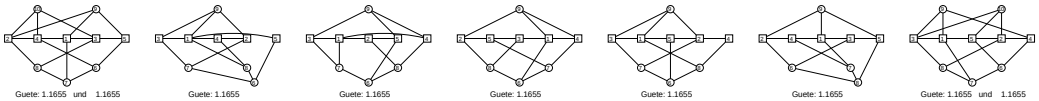
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693

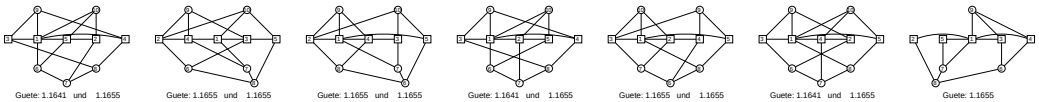


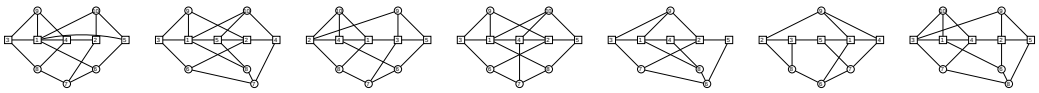
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693

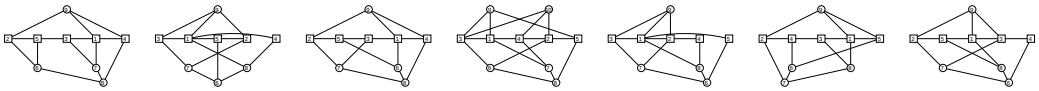

Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1693
Guete: 1.1655
Guete: 1.1655 und 1.1641

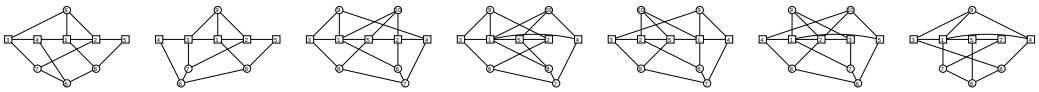

Guete: 1.1655
Guete: 1.1655
Guete: 1.1655 und 1.1631
Guete: 1.1641 und 1.1655
Guete: 1.1655 und 1.1641
Guete: 1.1655 und 1.1641
Guete: 1.1641 und 1.1655

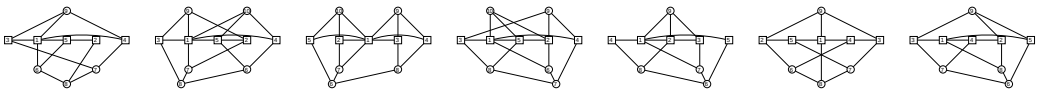

Guete: 1.1655 und 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655 und 1.1655

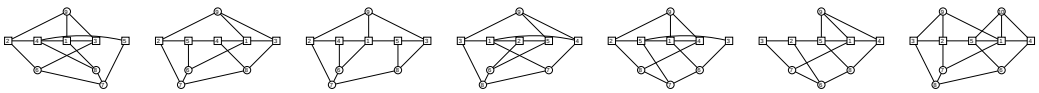

Guete: 1.1641 und 1.1655
Guete: 1.1655 und 1.1655
Guete: 1.1655 und 1.1655
Guete: 1.1641 und 1.1655
Guete: 1.1655 und 1.1655
Guete: 1.1641 und 1.1655
Guete: 1.1655

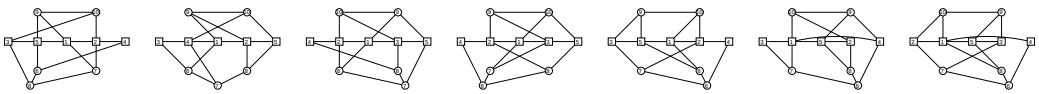

Guete: 1.1641 und 1.1655
Guete: 1.1641 und 1.1655
Guete: 1.1655 und 1.1655
Guete: 1.1641 und 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655 und 1.1655

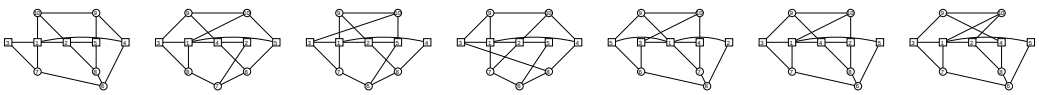

Guete: 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655 und 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655

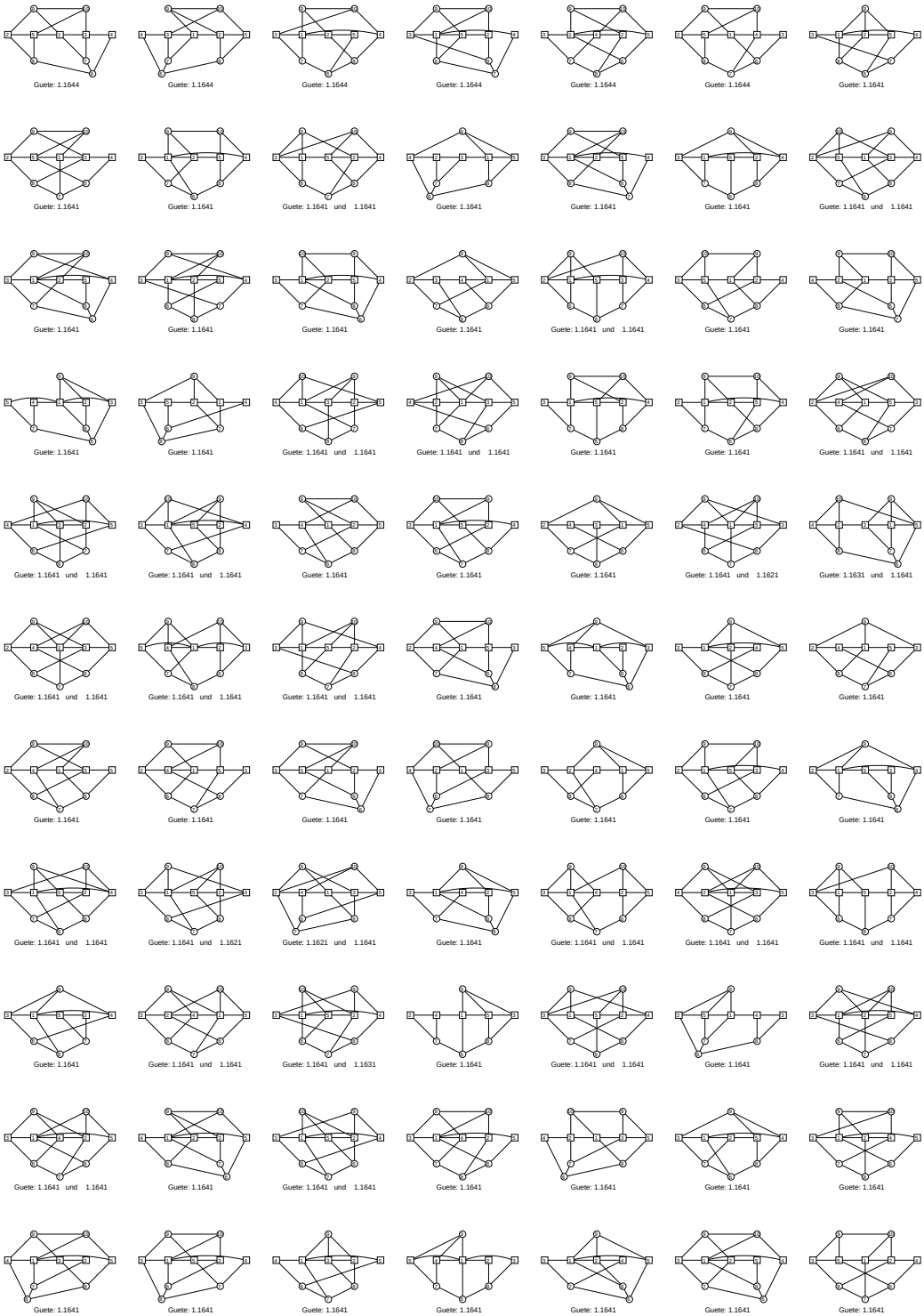

Guete: 1.1655
Guete: 1.1655
Guete: 1.1655 und 1.1655
Guete: 1.1641 und 1.1655
Guete: 1.1655 und 1.1655
Guete: 1.1641 und 1.1655
Guete: 1.1655

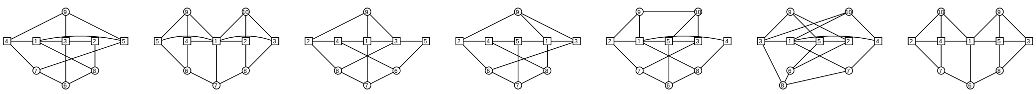

Guete: 1.1655
Guete: 1.1655 und 1.1641
Guete: 1.1641 und 1.1655
Guete: 1.1655 und 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655


Guete: 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655
Guete: 1.1655 und 1.1641

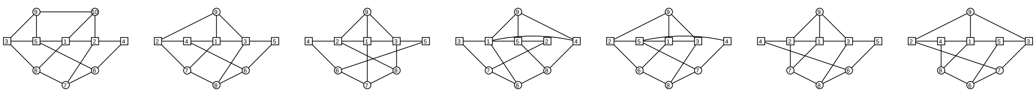

Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644


Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644

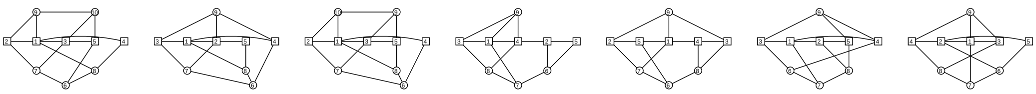

Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1644
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1631 und 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641 und 1.1621
Guete: 1.1621 und 1.1641
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641 und 1.1631
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641
Guete: 1.1641 und 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641
Guete: 1.1641



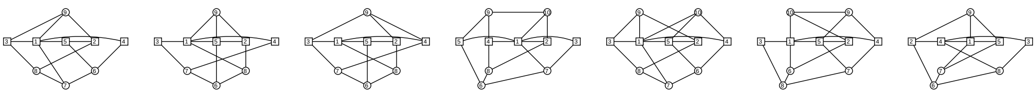
Guete: 1.1641 | Guete: 1.1641 und 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 und 1.1641 | Guete: 1.1641 und 1.1641



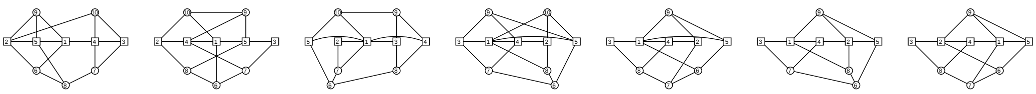
Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641



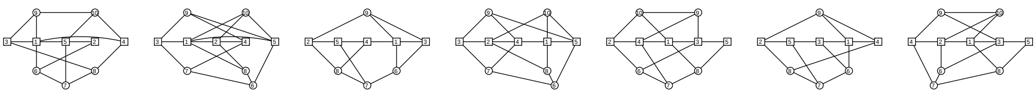
Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641



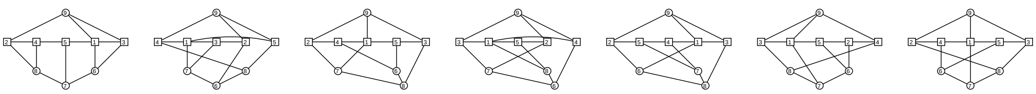
Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 und 1.1641 | Guete: 1.1641 | Guete: 1.1641



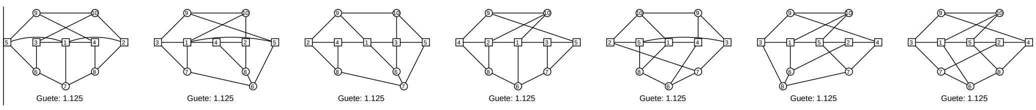
Guete: 1.1641 und 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641 und 1.1641 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641



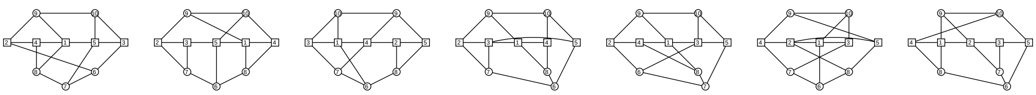
Guete: 1.1641 | Guete: 1.1641 und 1.1621 | Guete: 1.1641 | Guete: 1.1641 und 1.1631 | Guete: 1.1641 | Guete: 1.1641 | Guete: 1.1641



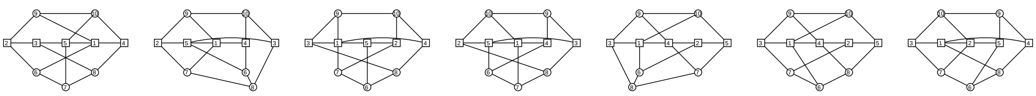
Guete: 1.1641 | Guete: 1.1631 | Guete: 1.1631 | Guete: 1.1621 | Guete: 1.1621 | Guete: 1.1621 | Guete: 1.1621



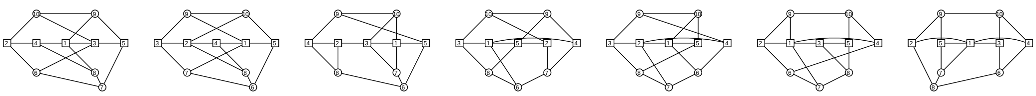
Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125



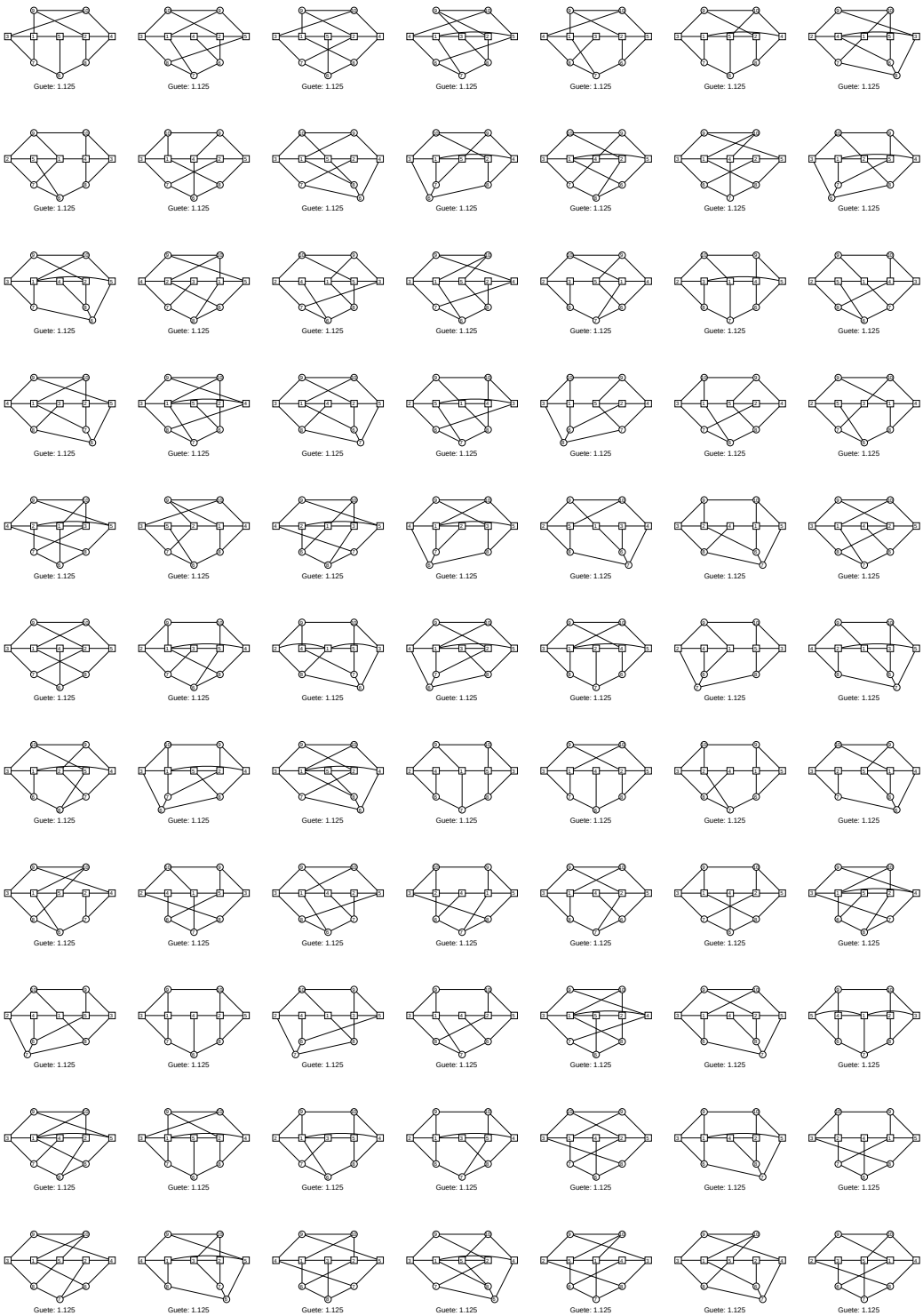
Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125

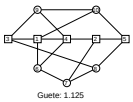


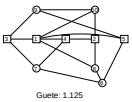
Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125

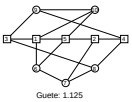


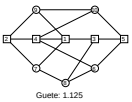
Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125 | Guete: 1.125

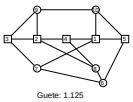

Guete: 1.125

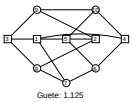

Guete: 1.125

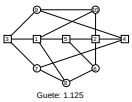

Guete: 1.125

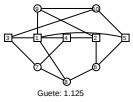

Guete: 1.125

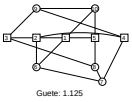

Guete: 1.125

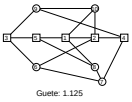

Guete: 1.125

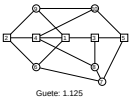

Guete: 1.125

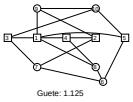

Guete: 1.125

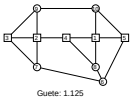

Guete: 1.125

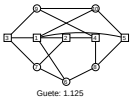

Guete: 1.125

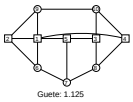

Guete: 1.125

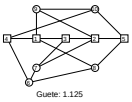

Guete: 1.125

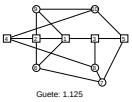

Guete: 1.125

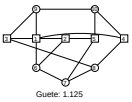

Guete: 1.125

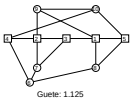

Guete: 1.125


Guete: 1.125


Guete: 1.125


Guete: 1.125


Guete: 1.125


Guete: 1.125

Literaturverzeichnis

[1] Aho, Alfred V. ; Hopcroft, John E.: *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974

[2] Arora, Sanjeev: Polynomial Time Approximation Schemes for Euclidean and other Geometric Problems. In: *Journal of the Association for Computing Machinery* 45 (1998), S. 753–782

[3] Arora, Sanjeev ; Lund, Carsten ; Motwani, Rajeev ; Sudan, Madhu ; Szegedy, Mario: Proof verification and the hardness of approximation problems. In: *Journal of the Association for Computing Machinery* 45 (1998), Nr. 3, S. 501–555

[4] Baudis, Gregor ; Gröpl, Clemens ; Hougardy, Stefan ; Nierhoff, Till ; Prömel, Hans J.: Approximating Minimum Spanning Sets in Hypergraphs and Polymatroids / Humboldt-Universität zu Berlin. 2000. – Forschungsbericht

[5] Berman, Piotr ; Fürer, Martin ; Zelikovsky, Alexander: Applications of the Linear Matroid Parity Algorithm to Approximating Steiner Trees. In: Grigoriev, D. (Hrsg.) ; Harrison, J. (Hrsg.) ; Hirsch, E. A. (Hrsg.): *Computer Science – Theory and Applications.* Springer, 2006 (Lecture Notes in Computer Science), S. 70–79

[6] Berman, Piotr ; Ramaiyer, Viswanathan: Improved approximations for the Steiner tree problem. In: *Journal of Algorithms* 17 (1994), S. 381–408

[7] Bern, Marshall W. ; Plassmann, Paul: The Steiner Problem with Edge Lengths 1 and 2. In: *Information Processing Letters* 32 (1989), S. 171–176

[8] Bollobás, Béla: *Random Graphs.* 2. Cambridge University Press, 2001

[9] Borchers, Al ; Du, Ding-Zhu: The k-Steiner ratio in graphs. In: *SIAM J. Comput.* 26 (1997), Nr. 3, S. 857–869

[10] Borůvka, Otakar: O jistém problému minimálním (about a certain minimal problem). In: *Práca Moravské Přirodvědecké Společnosti* 3 (1926), S. 37–58. – (Auf Tschechisch, Zusammenfassung auf Deutsch)

[11] Brazil, Marcus ; Nielsen, Benny K. ; Thomas, Doreen A. ; Winter, Pawel ; Wulff-Nilsen, Christian ; Zachariasen, Martin: *A Novel Approach to Phylogenetic Trees: d-dimensional geometric Steiner trees.* 2008. – erscheint in Networks

[12] Cheng, Yizong ; Church, George M.: Biclustering of Expression Data. In: *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, 2000, S. 93–103

[13] Chlebík, Miroslav ; Chlebíková, Janka: Approximation Hardness of the Steiner Tree Problem on Graphs. In: *Algorithm Theory - SWAT 2002, 8th Scandinavian Workshop on Algorithm Theory* Bd. 2368, Springer, 2002 (Lecture Notes in Computer Science), S. 170–179

[14] Cyvin, S. J. ; Brunvoll, J. ; Cyvin, B. N.: Enumeration of constitutional isomers of polyenes. In: *Journal of Molecular Structure: THEOCHEM* 357 (1995), Nr. 3, S. 255–261

[15] Dawande, Milind ; Keskinocak, Pinar ; Swaminathan, Jayashankar M. ; Tayur, Sridhar: On bipartite and multipartite clique problems. In: *Journal of Algorithms* 41 (2001), Nr. 2, S. 388–403

[16] Diané, Mamadi ; Plesník, Ján: An integer programming formulation of the Steiner problem in graphs. In: *Mathematical Methods of Operations Research* 37 (1993), Nr. 1, S. 107–111

[17] Dijkstra, Edsger W.: A note on two problems in connexion with graphs. In: *Numerische Mathematik* 1 (1959), S. 269–271

[18] Dreyfus, S. E. ; Wagner, R. A.: The Steiner problem in graphs. In: *Networks* 1 (1972), S. 195–207

[19] Erdős, Paul ; Spencer, Joel H.: *Probabilistic Methods in Combinatorics.* Academic Press, Inc., 1974 (Probability and Mathematical Statistics – A Series of Monographs and Textbooks)

[20] Erdős, Paul ; Rényi, Alfréd: On the Evolution of Random Graphs. In: *Publ. Math, Inst. Hung. Acad. Sci.* 5 (1960), S. 17–61

[21] Erdős, Paul ; Simonovits, Miklós: A Limit Theorem in Graph Theory. In: *Studia Scientiarium Mathematicarum Hungarica* 1 (1966), S. 51–57

[22] Erdős, Paul ; Stone, A. H.: On the Structure of Linear Graphs. In: *Bulletin of the American Mathematical Society* 52 (1946), S. 1087–1091

[23] Feder, Tomás ; Motwani, Rajeev: Clique Partitions, Graph Compression and Speeding-Up Algorithms. In: *Journal of Computer and System Sciences* 51 (1995), Nr. 2, S. 261–272

[24] Feige, Uriel ; Kogan, Shimon: Hardness of approximation of the Balanced Complete Bipartite Subgraph problem / The Wizmann Institute of Science. 2004 (MCS-04-04). – Forschungsbericht

[25] Fuchs, B. ; Kern, W. ; Molle, D. ; Richter, S. ; Rossmanith, P. ; Wang, X.: Dynamic Programming for Minimum Steiner Trees. In: *Theory of Computing Systems* 41 (2007), Nr. 3, S. 493–500

[26] Garey, Michael R. ; Johnson, David S.: *Computers and Intractability.* W. H. Freeman and Company, 1979

[27] Gilbert, E. N. ; Pollak, H. O.: Steiner minimal trees. In: *SIAM Journal on Applied Mathematics* 16 (1968), Nr. 1, S. 1–29

[28] Gröpl, Clemens ; Hougardy, Stefan ; Nierhoff, Till ; Prömel, Hans J.: Approximation algorithms for the Steiner tree problem in graphs. In: Cheng, Xiuzhen (Hrsg.) ; Du, Ding-Zhu (Hrsg.): *Steiner Trees in Industry.* Kluwer Academic Publishers, 2001, S. 235–279

[29] Gröpl, Clemens ; Hougardy, Stefan ; Nierhoff, Till ; Prömel, Hans J.: Lower bounds for approximation algorithms for the Steiner tree problem. In: *Proceedings WG 2001* Bd. 2204, Springer Verlag, 2001 (Lecture Notes in Computer Science), 217–228

[30] Guo, Longjiang ; Wu, Weili ; Wang, Feng ; Thai, My: An Approximation for Minimum Multicast Route in Optical Networks with Nonsplitting Nodes. In: *Journal of Combinatorial Optimization* 10 (2005), Nr. 4, S. 391–394

[31] Hanke, Sven ; Kirchner, Stefan: *Neue untere Schranken für den Relativen Greedy Algorithmus und eine optimierte Implementation.* Humboldt-Universität zu Berlin, Oktober 2002. – Studienarbeit

[32] Håstad, Johan: Clique is Hard to Approximate Within $n^{1-\epsilon}$. In: *Acta Mathematica* 182 (1999), S. 105–142

[33] Hochbaum, Dorit S.: Approximating clique and biclique problems. In: *Journal of Algorithms* 29 (1998), Nr. 1, S. 174–200

[34] Hoory, Shlomo ; Linial, Nathan ; Wigderson, Avi: Expander Graphs and their Applications. In: *Bulletin (new series) of the American Mathematical Society* 43 (2006), Nr. 4, S. 439–561

[35] Hopcroft, John E. ; Karp, Richard M.: An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. In: *SIAM J. Comput.* 2 (1973), Nr. 4, S. 225–231

[36] Hougardy, Stefan ; Kirchner, Stefan: Lower Bounds for the Relative Greedy Algorithm for Approximating Steiner Trees. In: *Networks* 47 (2006), Nr. 2, S. 111–115

[37] Hougardy, Stefan ; Prömel, Hans J.: A 1.598 Approximation Algorithm for the Steiner Problem in Graphs. In: *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1999, S. 448–453

[38] Ji-Xing ; Shi-Bao, Zheng: *New Lower Bounds for Approximation Algorithm for the Steiner Tree Problem*. 2007. – unveröffentlicht

[39] Johnson, David S.: The NP-completeness column: An ongoing guide. In: *Journal of Algorithms* 8 (1987), Nr. 5, S. 438–448

[40] Jukna, Stasys: *Extremal Combinatorics*. Springer, 2001 (Texts in Theoretical Computer Science)

[41] Karp, Richard: Reducibility among combinatorial problems. In: Miller, R. (Hrsg.) ; Tatcher, J. (Hrsg.): *Complexity of Computer Computations*. Plenum Press, 1972, S. 85–103

[42] Karpinski, Marek ; Zelikovsky, Alexander Z.: New approximation algorithms for the Steiner tree problems. In: *Journal of Combinatorial Optimization* 1 (1997), S. 47–65

[43] Kaufmann, Michael ; Gao, Shaodi ; Thulasiraman2, K.: On Steiner minimal trees in grid graphs and its application to VLSI routing. In: *Algorithms and Computation* Bd. 834, Springer Verlag, 1994 (Lecture Notes in Computer Science), S. 351–359

[44] Khot, Subhash ; Regev, Oded: Vertex cover might be hard to approximate to within $2-\epsilon$. In: *Journal of Computer and System Sciences* 74 (2008), Nr. 3, S. 335–349

[45] Kirchner, Stefan: *Neue untere Schranken für den Relativen Greedy Algorithmus*. Humboldt-Universität zu Berlin, 2002. – Studienarbeit

[46] Kleinberg, Jon: An Impossibility Theorem for Clustering. In: *Advances in Neural Information Processing Systems* Bd. 15, 2002

[47] Kövari, Thomas ; Sós, Vera T. ; Turán, Paul: On a Problem of Zarankiewicz. In: *Colloq. Math.* 3 (1954), S. 50–57

[48] Kruskal, Joseph: On the shortest spanning subtree and the traveling salesman problem. In: *Proceedings of the American Mathematical Society* 7 (1956), S. 48–50

[49] Madeira, Sara C. ; Oliveira, Arlindo L.: Biclustering Algorithms for Biological Data Analysis: A Survey. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 1 (2004), Nr. 1, S. 24–45

[50] Mehlhorn, Kurt: A faster approximation algorithm for the Steiner problem in graphs. In: *Information Processing Letters* 27 (1988), S. 125–128

[51] Meilă, Marina: Comparing clusterings: an axiomatic view. In: *ICML '05: Proceedings of the 22nd international conference on Machine learning*, ACM, 2005, S. 577–584

[52] Mishra, Nina ; Ron, Dana ; Swaminathan, Ram: On Finding Large Conjunctive Clusters. In: *COLT*, 2003, S. 448–462

[53] Peeters, René: The maximum edge biclique problem is NP-complete. In: *Discrete Applied Mathematics* 131 (2003), Nr. 3, S. 651–654

[54] Prim, Robert C.: Shortest connection networks and some generalisations. In: *Bell System Technical Journal* 36 (1957), S. 1389–1401

[55] Prömel, Hans J. ; Steger, Angelika: A new approximation algorithm for the Steiner tree problem with performance ratio 5/3. In: *Journal of Algorithms* 36 (2000), S. 89–101

[56] Puolamäki, Kai ; Hanhijärvi, Sami ; Garriga, Gemma C.: *An Approximation Ratio for Biclustering.* 2007. – unveröffentlicht

[57] Robins, Gabriel ; Zelikovsky, Alexander: Tighter Bounds for Graph Steiner Tree Approximation. In: *SIAM J. Discret. Math.* 19 (2005), Nr. 1, S. 122–134

[58] Rudin, Walter: *Analysis.* 2. Oldenbourg, 2002

[59] Takahashi, H. ; Matsuyama, A.: An approximate solution for the Steiner problem in graphs. In: *Mathematica Japonica* 24 (1980), S. 573–577

[60] Thimm, Martin: On the Approximability of the Steiner Tree Problem. In: *Theoretical Computer Science* 295 (2003), S. 387–402

[61] Thimm, Martin: *Algorithmen im Wirkstoffdesign*, Humboldt-Universität zu Berlin, Diss., 2005

[62] Watts, Valerie L.: Fractional Biclique Covers and Partitions of Graphs. In: *The Electronic Journal of Combinatorics* 13 (2006), Nr. 1, S. R74

[63] Winter, Pawel: Steiner problem in networks: A survey. In: *Networks* 17 (1987), S. 129–167

[64] Zelikovsky, Alexander Z.: An 11/6-approximation algorithm for the network Steiner problem. In: *Algorithmica* 9 (1993), S. 463–470

[65] Zelikovsky, Alexander Z.: Better approximation algorithms for the network and Euclidean Steiner tree problems / University of Virginia. 1996 (CS-96-06). – Forschungsbericht

[66] Zuckerman, David: Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. In: *Theory of Computing* 3 (2007), Nr. 6, S. 103–128

Danksagung

Zunächst möchte ich mich bei Professor Hans Jürgen Prömel bedanken, der mir die Möglichkeit gab, als Mitarbeiter an seinem Lehrstuhl und am MATHEON zu arbeiten.

Mein zweiter Dank gilt Stefan Hougardy für die gute Betreuung in dieser Zeit. Außerdem hörte ich bei ihm vor langer Zeit die Vorlesungsreihe „Graphen und Algorithmen“, die Ausgangspunkt für mich war, mich näher mit Diskreter Mathematik und insbesonderen Steinerbäumen zu befassen.

Bei Professor Anand Srivastav möchte ich mich dafür bedanken, dass er sich als Fremdgutachter bereit erklärt hat.

Mein weiterer Dank gilt Mathias Schacht für fruchtbare Gespräche über Bicliquen. Zuletzt möchte ich mich bei Dirk Schlatter und Valentin Ziegler bedanken, dass sie Teile dieser Arbeit gelesen und wertvolle Hinweise gegeben haben.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet zu haben.

Ich erkläre, dass ich die Arbeit erstmalig und nur an der Humboldt-Universität zu Berlin eingereicht habe und mich nicht anderwärts um einen Doktorgrad beworben habe. In dem Promotionsfach besitze ich keinen Doktorgrad.

Der Inhalt der dem Verfahren zugrunde liegenden Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II, veröffentlicht im Amtlichen Mitteilungsblatt Nr. 34/2006, ist mir bekannt.

Berlin, den 25. März 2008

Stefan Kirchner